The Dissertation Committee for Kurt Mauro Dresner
certifies that this is the approved version of the following dissertation:

Autonomous Intersection Management

Committee:

Peter Stone, Supervisor

Benjamin Kuipers

Bruce Porter

Manuela Veloso

S. Travis Waller

Autonomous Intersection Management

by

Kurt Mauro Dresner, B.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2009

For my mother.

Acknowledgments

This thesis, as well as the work it represents, would not have been possible without the continuing insight, inspiration, and tireless efforts of my advisor, Peter Stone. Because of his instruction and his example, I am a better researcher, a better writer, and a better scientist. I am also grateful to my committee members, Ben Kuipers, Travis Waller, Bruce Porter, and Manuela Veloso who provided useful feedback and asked tough questions that helped me make this thesis significantly stronger. By supporting the Learning Agents Research Group, the following agencies allowed me to focus on research, for which I am grateful: NSF, the Federal Highway Administration, General Motors, and DARPA.

I would also like to thank several individuals for their direct contributions to the work in this thesis. First, Mark VanMiddlesworth, for finding several big bugs in my code, developing the first version of the Vehicle-to-Vehicle protocol, and helping write and formalize the current version. Second, Justin Brickell for discussing and analyzing my algorithms for correctness and rigor. Third, Tarun Nimmagadda, for creating the first mixed simulation using my simulator.

In addition to direct contributions, I have enjoyed the conversation and camaraderie of many members of my lab and research group, who have provided inspiration and valuable insights. These people include Greg Kuhlmann, Dan Stronger, Nate Kohl, Nick Jong, as well as other members of "The Aibo Lab."

Finally, I'd like to recognize my family. My mother has had amazing confi-

dence in me and insisted that I attack challenges head-on. My father has continually encouraged me to stay focused and has helped make my life as a graduate student a lot more comfortable. My beautiful daughter Keziah, who came into our life recently, has been a source of joy and motivation to get graduated. Most significantly, I'd like to thank my wonderful wife Kendra, without whose unflagging support, encouragement, patience, and love, I surely would not have completed this thesis.

Kurt Mauro Dresner

The University of Texas at Austin
December 2009

Autonomous Intersection Management

Publication No. _____

Kurt Mauro Dresner, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Peter Stone

Artificial intelligence research is ushering in an era of sophisticated, mass-market transportation technology. While computers can fly a passenger jet better than a human pilot, people still face the dangerous yet tedious task of driving. Intelligent Transportation Systems (ITS) is the field focused on integrating information technology with vehicles and transportation infrastructure. Recent advances in ITS point to a future in which vehicles handle the vast majority of the driving task. Once autonomous vehicles become popular, interactions amongst *multiple* vehicles will be possible. Current methods of vehicle coordination will be outdated. The bottleneck for efficiency will no longer be drivers, but the mechanism by which those drivers' actions are coordinated.

Current methods for controlling traffic cannot exploit the superior capabili-

ties of autonomous vehicles. This thesis describes a novel approach to managing autonomous vehicles at intersections that decreases the amount of time vehicles spend waiting. Drivers and intersections in this mechanism are treated as autonomous agents in a multiagent system. In this system, agents use a new approach built around a detailed communication protocol, which is also a contribution of the thesis. In simulation, I demonstrate that this mechanism can significantly outperform current intersection control technology—traffic signals and stop signs.

This thesis makes several contributions beyond the mechanism and protocol. First, it contains a distributed, peer-to-peer version of the protocol for low-traffic intersections. Without any requirement of specialized infrastructure at the intersection, such a system would be inexpensive and easy to deploy at intersections which do not currently require a traffic signal. Second, it presents an analysis of the mechanism's safety, including ways to mitigate some failure modes. Third, it describes a custom simulator, written for this work, which will be made publicly available following the publication of the thesis. Fourth, it explains how the mechanism is "backward-compatible" so that human drivers can use it alongside autonomous vehicles. Fifth, it explores the implications of using the mechanism at multiple proximal intersections. The mechanism, along with all available modes of operation, is implemented and tested in simulation, and I present experimental results that strongly attest to the efficacy of this approach.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Few concepts, if any, embody the goals and aspirations of artificial intelligence as well as fully autonomous robots. Countless films and stories have been made that focus on a future filled with such humanoid agents which, when not violently overthrowing their human masters, complete menial tasks, run errands, or carry out jobs that humans cannot or will not do. However, machines that sense, think about, and take actions in the real world around us are no longer just the stuff of science fiction and fantasy. Research initiatives like Robocup [Noda *et al.*, 2006] and the DARPA Grand Challenge [DARPA, 2007a] have shown that current AI can produce autonomous, embodied, competent agents for complex tasks like playing soccer or navigating the Nevada Desert, respectively. While certainly no small feat, traversing a barren desert devoid of pedestrians, narrow lanes, and multitudes of other fast-moving vehicles is not a typical daily task for humans. As Gary Bradski, a researcher at Intel Corp. said following the successful completion of the 2005 Grand Challenge by "Stanley," a modified Volkswagen Touareg, "Now we need to teach them how to drive in traffic" [Johnson, 2005].

In modern urban settings, automobile traffic and collisions lead to endless frustration as well as significant loss of life, property, and productivity. A 2004 study

of 85 U.S. cities by researchers at Texas A&M University estimated the annual time spent waiting in traffic at 46 hours—more than a whole work week—per capita, up from 16 hours in 1982 [Texas Transportation Institute, 2004]. Americans burn approximately 5.6 billion gallons of fuel each year simply idling their engines. All told, the annual financial cost of traffic congestion has swollen from $14 billion to more than $63 billion (in 2002 US dollars) in this period. The cost of all the wasted time and fuel due to congestion pales in comparison to the costs associated with automobile collisions. A report by the National Highway Traffic Safety Administration (NHTSA) puts the annual societal cost of automobile collisions in the U.S. at $230 billion [National Highway Traffic Safety Administration, 2002].

Fully autonomous vehicles may be able to spare us much, if not nearly all of these costs. An autonomous driver agent can much more accurately judge distances and velocities, attentively monitor its surroundings, and react instantly to situations that would leave a (relatively) sluggish human driver helpless. Furthermore, an autonomous driver agent will not get sleepy, impatient, angry, or drunk. Alcohol, speeding, and running red lights are the top three causes of automobile collision fatalities. Autonomous driver agents — properly programmed — would eliminate all three.

A fully autonomous vehicle that will drive in traffic will have to do everything from obeying the speed limit and staying in its lane to detecting and tracking pedestrians or choosing the best route to the mall. While this is certainly a complex task, advances in artificial intelligence, and more specifically, Intelligent Transportation Systems (ITS)[Bishop, 2005], suggest that it may soon be a reality. Cars can already be equipped with features of autonomy such as adaptive cruise control, GPS-based route planning [Rogers *et al.*, 1999; Schonberg *et al.*, 1995], and autonomous steering [Pomerleau, 1993; Reynolds, 1999]. Some current production vehicles even sport these features. DaimlerBenz's Mercedes-Benz S-Class has an adaptive cruise

control system that can maintain a safe following distance from the car in front of it, and will apply extra braking power if it determines that the driver is not braking hard enough. Both Toyota and BMW are currently selling vehicles that can parallel park completely autonomously, even finding a space in which to park without driver input.

Autonomous vehicles are coming. In this thesis, I describe a well-defined multiagent framework and show that it can dramatically improve the safety and efficiency of roadways with autonomous vehicles, specifically at intersections.

1.1 Multiagent Systems

As autonomous vehicles become ubiquitous, the possibility of autonomous interactions among multiple vehicles becomes an interesting issue. Multiagent Systems (MAS) is the subfield of AI that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors [Wooldridge, 2002]. Automobile traffic as it stands today is a vast multiagent system involving millions of heterogeneous agents: commuters, truck drivers, pedestrians, cyclists, and even traffic-directing police officers. The mechanism that coordinates the behavior of these agents is a complex conglomeration of laws, signs, and signalling systems that vary slightly from state to state and widely from country to country. The mechanism is designed to work closely with the agents — the humans — that populate the multiagent system. Traffic signals leave time in between green signals to allow slower or perhaps impatient drivers to clear intersections. Street signs are bright colors to make them easier to see and simple (i.e. they don't convey much information) to make them easy to understand. Drivers must maintain a sufficient following distance to make up for slow reaction times. Speed limits ensure that humans can process all the necessary information about the position and velocities of other vehicles in order to operate their vehicles

safely. Safety buffers of myriad sorts are built into almost every part to compensate for the limitations of humans.

The first autonomous vehicles will undoubtedly need to work within this system. Processing-intensive vision algorithms will identify and extract semantic information from signs and signals, special subroutines will ensure that the vehicles do not exceed the speed limit, and in the middle of the night, with not another moving vehicle for blocks, an autonomous vehicle will come to a stop at a red light. However, once most vehicles are autonomous and the limitations are eliminated, it does not make sense to use a mechanism designed to control fundamentally different agents — it will be inefficient, both in terms of processing power and getting vehicles to their destinations.

Replacing this soon-to-be-outdated mechanism is inherently a multiagent challenge for several reasons. First, there are no viable single-agent solutions; one computer cannot handle all the vehicles in the world. Second, with vehicles constantly entering and leaving countries, states, cities, and towns, any solution will have to be flexible and distributed. Third, the different agents have separate, and sometimes conflicting objectives. As with human-driven vehicles, autonomous vehicles will act in their own self-interest, attempting to minimize travel time, distance, and fuel use. Other agents may aim to maximize social welfare, minimizing these quantities for the average vehicle. Finally, even if a single computer could control a city's worth of traffic, it would be a very sensitive point of failure.

1.2 Intersections

On the open road, automobiles can be more or less completely autonomous. Furthermore, there is little need for more than a simple reactive behavior that keeps the vehicle in the lane, maintains a reasonable distance from other vehicles, and avoids obstacles. Even lane changing can be safely and efficiently accomplished by

an autonomous vehicle [Hatipo *et al.*, 1997]. Open-road driving is more or less a solved problem. The problem itself is not too difficult: there are no pedestrians or cyclists and vehicles travel in the same direction at similar velocities; relative movement is smooth and rare.

Intersections are a completely different story. Vehicles are constantly crossing paths, frequently in different directions. A vehicle approaching an intersection can quickly find itself in a situation in which a collision is unavoidable, even when it has acted optimally. Traffic statistics support the sensitive nature of intersections. Vehicle collisions at intersections account for anywhere between 25% and 45% of all collisions. As intersections make up a very small portion of the roadway, this is a wildly disproportionate amount. Collisions at intersections tend to involve cars traveling in different directions, and thus they frequently result in greater injury and damage. Most modern-day intersections are controlled with traffic signals or stop signs, the former usually reserved for larger, busier intersections. At the busiest of intersections—freeway interchanges—large, extremely expensive cloverleaf junctions are built.

With the vastly improved precision control and sensing that autonomous vehicles will offer, there must be a more efficient and safe way to manage intersections. Imagine the scenario in which an autonomous vehicle stops at a red signal in the middle of the night with no other vehicles nearby. At the very least, the vehicle should be able to communicate its presence to the intersection, which can verify that no other vehicles are nearby, and turn the signal green for the stopped vehicle. In a more ambitious implementation, the intersection could turn the signal green preemptively, obviating the stop altogether.

1.3 This Thesis

Motivated by the preceding discussion, this thesis answers the following question:

To what extent and how can a multiagent intersection control mechanism take advantage of the capabilities of autonomous vehicles in order to make automobile travel safer and faster?

In order to answer this question, this thesis is organized according to the following subgoals, each of which is a contribution of the thesis.

1. **Problem definition**

   First, this thesis contributes a careful and specific problem definition. This definition includes a set of desiderata describing a successful solution, including safety, efficiency, and feasibility requirements.

2. **Performance metric**

   A multitude of solutions may exist for any given problem. In order to compare possible solutions (and find the best one), this thesis introduces the concept of *delay*—the increase in travel duration caused by the intersection.

3. **Novel intersection control mechanism**

   Today's intersection control mechanisms were designed to work with humans. This thesis presents a solution designed from the ground up to take advantage of the special abilities of autonomous vehicles. The solution is based on a reservation paradigm, in which vehicles "call ahead" to reserve space-time in the intersection.

4. **Detailed protocol**

   A multiagent system is defined by the interactions of its agents. Because this thesis aims to create a multiagent solution, it must specify how the agents will be expected to behave with respect to one another, including exactly how they will communicate. This specification takes the form of a detailed

protocol, complete with message types and fields, an intended semantics, and interaction rules governing expected message responses.

5. **Custom simulator**

   Because the vast majority of computer-assisted traffic research focuses on improving or studying current methods of traffic control, existing simulators do not give enough flexibility or control to specify fundamentally different intersection control mechanisms without extensive source code understanding and alteration. Additionally, many simulators model in much more detail than is necessary for the purposes of this research. This thesis research thus includes an extensive implementation component, including not only the framework for simulating vehicles, but also the driver agents, communication protocol, and control mechanisms.

6. **Agent algorithms**

   While agent interactions define a multiagent system, the behaviors of the agents themselves often most directly contribute to the performance of the system as a whole. The protocol makes certain guarantees about the system (e.g. safety, robustness under communication failure), but it also defines very large strategy spaces for the agents. The main technical contribution of this thesis is an extensive exploration of these spaces. Strategies for all agents in the system are examined, including adaptive strategies.

7. **Empirical evaluation**

   Before a new mechanism can be considered for deployment in the real world, it must perform quantifiably better than both current methods and ideally all other possible solutions. This thesis provides detailed empirical results, in a variety of settings, including some in which human drivers are present.

8. **Feasibility analysis for implementation and deployment**

For such a system to be useful, it must be realizable in some form. Regardless of how well it performs, if it is prohibitively expensive or complicated to deploy, it will remain only a concept. In addition to a discussion of the issues and challenges associated with putting the system into operation, this thesis provides a transitional method by which the system can be smoothly and incrementally deployed.

The remainder of this thesis is organized as follows. Chapter 2 introduces the main concepts behind the Autonomous Intersection Management (AIM) system. The protocol by which this system operates is specified in Chapter 3. Chapters 4 and 5 describe our implementations of the intersection manager and driver agents, respectively. In Chapter 6, I introduce the custom simulator created for evaluating the AIM system. I present our experimental results on the base AIM system in Chapter 7. Chapter 8 presents the first extension to the base AIM system, that allows the system to be used with a mix of autonomous and human drivers, with an efficiency penalty that shrinks as the proportion of human drivers decreases. Chapter 9, contains a failure-mode analysis that explores some of the worst-case scenarios and how to mitigate or prevent them. The second major extension, which allows the AIM system to function safely at networks with multiple equipped intersections, is presented in Chapter 10. In Chapter 11, I discuss other work in a variety of fields that is either directly related or focused on requisite technologies. Finally, Chapter 12 concludes.

Chapter 2

The Autonomous Intersection Management System

This chapter introduces the core idea of the thesis, namely reservation-based intersection management. This chapter does not give specific details on implementation and evaluation—those can be found in Chapters 3, 4, 5, and 7—but rather provides a high-level overview of the challenges and concepts used to address those challenges. First, it lays out the desiderata we have established for any system that would replace our current system of intersection management for human-driven vehicles. Then it describes the types of agents and the multiagent systems those agents will form as part of the intersection control mechanism.

2.1 Desiderata

Replacing modern intersection control with a robust, multiagent framework is a complex, multi-part problem. In order to choose directions in which to focus my research as well as establish a set of criteria by which to judge such a framework, this section enumerates an important set of properties I believe any intersection control

mechanism for autonomous vehicles should have.

2.1.1 Autonomy

Each vehicle should be an autonomous agent. If the entire mechanism were centrally controlled, it would be more susceptible to single point failure, require massive amounts of computational power, and would exert unnecessary control over vehicles in situations where they are perfectly capable of controlling themselves.

2.1.2 Low Communication Complexity

By keeping the number of messages and amount of information transmitted to a minimum, the system can afford to put more communication reliability measures in place. Furthermore, each vehicle, as an autonomous agent, may have privacy concerns which should be respected. Keeping the communication complexity low will also make the system more scalable.

2.1.3 Sensor Model Realism

Each agent should have access only to sensors that are available with current-day technology. The mechanism should not rely on fictional sensor technology that may never materialize.

2.1.4 Protocol Standardization

The mechanism should employ a simple, standardized protocol for communication between agents. Without a standardized protocol, each agent would need to understand the internal workings of every agent with which it interacted. This requirement would forbid the introduction of new agents into the system. An open, standardized protocol would make adoption of the system easier and simpler for private vehicle manufacturers.

2.1.5 Deadlock/Starvation Avoidance

Deadlocks and starvation should not occur in the system. Any vehicle approaching an intersection should eventually make it through, even if it is better for the rest of the agents to leave that vehicle stranded.

2.1.6 Incremental Deployability

The system should be incrementally deployable, in two senses. First, it should be possible to set up selected intersections to use the system, and then slowly expand to other intersections as needed. Second, the system should function even with few or no autonomous vehicles. At any stage of deployment, be it an increase in the proportion of autonomous vehicles or number of equipped intersections, overall performance of the system should improve, and there should be a benefit to early adopters. At no point should there exist a net disincentive to continue deploying the system.

2.1.7 Safety

Excepting for gross vehicle malfunction or extraordinary circumstances (natural disasters, etc.), as long as they follow the protocol, vehicles should never collide in the intersection. Note that no stronger guarantee is possible — as with modern mechanisms, a suicidal human driver can always steer a vehicle into oncoming traffic. Furthermore, the system should be safe in the event of total communication failure. If messages are dropped or corrupted, the safety of the system should not be compromised. It is impossible to prevent all negative effects due to communication failures, but those negative effects should be isolated to efficiency. If a packet gets dropped, it can make someone arrive 10 seconds later at their destination, but it should not cause a collision.

2.1.8 Efficiency

Vehicles should get across the intersection and on their way in as little time as possible. To quantify efficiency, we introduce *delay*, defined as the amount of additional travel time incurred by the vehicle as the result of passing through the intersection.

2.2 New Traffic, New Management

Of the desiderata, modern-day traffic signals and stop signs completely satisfy all but the last one. While many accidents take place at intersections governed by traffic signals, these accidents are rarely, if ever, the fault of the traffic signal system itself, but rather that of the human drivers. However, traffic signals and stop signs are not very efficient. Not only do vehicles traversing intersections equipped with these mechanisms experience large delays, but the intersections themselves can only manage a somewhat limited amount of traffic. Any stretch of open road can accommodate a certain level of traffic at a given velocity. Obviously, the capacity of an intersection involving such a road is bounded above by the capacity of the road. The capacity of traffic signals and stop signs is much less than that of the roads that feed into them. Much of this inefficiency is due to large allowances for the inadequacies of human drivers: slow reaction times, poor perception, and tendency to drive while impaired. With the introduction of computerized drivers that do not possess these deficiencies, the large allowances required for human drivers are no longer necessary. Removing these allowances enables the intersection control mechanism presented in this thesis to exceed the efficiency of traffic signals and stop signs without sacrificing any of the other properties.

2.2.1 Agents

Keeping the desiderata in mind, we developed a multiagent approach to get vehicles through intersections more efficiently. This approach involves two classes of agents: *driver agents* that control the vehicles, and *arbiter* agents called *intersection managers* at some intersections. When an intersection manager is present and coordinating the traversal of the intersection, it is called a Vehicle-To-Infrastructure or Vehicle-To-Intersection (V2I) system. When no intersection manager is present, the vehicles must operate in a completely distributed Vehicle-To-Vehicle (V2V) manner.

Driver Agents

An agent is an entity that can sense its environment and take actions that have an effect on that environment. In general, a driver agent can be any agent that drives a vehicle, including a human driver. However, in this thesis, when I refer to a driver agent, I usually mean a computer program that controls some amount of the driving of a vehicle—at the very least, the acceleration and steering, but potentially route or even destination choice. In some of the future work discussed in this thesis, I mention *hybrid driver agents*, which include both a computer program and a human working together to operate the vehicle.

Intersection Managers

Analagous to the definition of a driver agent, an intersection manager is simply an agent that controls access to an intersection. An intersection with an intersection manager is referred to as a *managed* intersection, while one without an intersection manager is *unmanaged*. While not every intersection requires a dedicated intersection manager, having an intersection manager allows vehicle to coordinate their interactions with a higher degree of precision, resulting in more throughput and less wasted time.

2.2.2 V2I

In modern-day traffic systems, different intersections may be controlled via different mechanisms. For example, at an intersection with extremely little traffic, there may be no explicit management whatsoever. Drivers may be expected to watch for other vehicles and in the rare even of encountering one, work out which will yield to the other by visual cues such as waving the other driver on. As intersections become more heavily used, a two- or four-way stop may be created. At even busier intersections, more sophisticated systems that do not always require a stop such as traffic signals or a roundabout may be used.

Analagously, autonomous intersections with heavier traffic require more explicit management. These scenarios are called "Vehicle-to-Intersection" or "Vehicle-to-Infrastructure" (V2I) scenarios. In V2I scenarios, an intersection manager is present at the intersection, and it is the responsibility of this intersection manager to resolve the conflicts between vehicles' trajectories.

The Reservation Paradigm

The driver agents "call ahead" and attempt to reserve a block of space-time in the intersection. The intersection manager decides whether to grant or reject requested reservations according to an *intersection control policy*. If the request is granted, the driver agent may proceed through the intersection in accordance with the reservation. If the request is rejected, the driver agent must make another request, and cannot cross until one of its requests is granted. Figure 2.1 shows one interaction between a driver agent and an intersection manager. The system functions analagously to a human attempting to make a reservation at a hotel — the potential guest specifies when he or she will be arriving, how much space is required, and how long the stay would be; the human reservation agent determines whether or not to grant the reservation, according to the hotel's reservation policy. Just as the guest

does not need to understand the hotel's decision process, the driver agents should not require any knowledge of how the policy the intersection manager uses to make its decision. In addition to confirming the request made by the driver agent, the intersection manager may also respond with counter-offers.

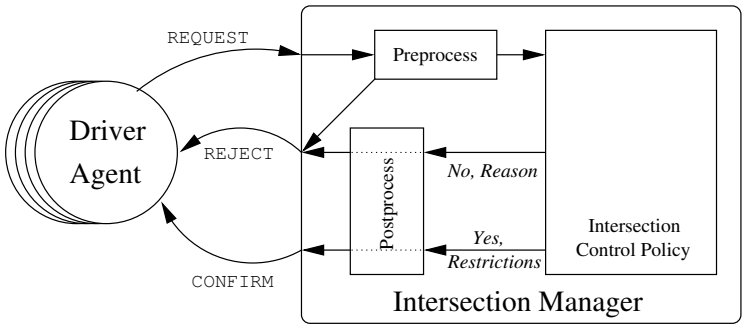


Figure 2.1: One of the driver agents attempts to make a reservation. The intersection manager responds based on the decision of an intersection control policy.

Technological Basis

In order for a driver agent to communicate with an intersection early enough, it may have to transmit messages as far as 200 meters, which at 25 m/s (approximately 56 miles per hour) is 8 seconds before reaching the intersection. By placing transmitters and receivers along the roadway approaching the intersection, even a system created from off-the-shelf wireless networking components (802.11b/g/n) would likely be sufficient for decent performance. However, automotive equipment manufacturers are already building application-specific hardware, such as the Denso Corporation's Wireless Safety Unit (WSU), which is a small, embedded machine that uses Digital Short-Range Communications (DSRC)—part of the IEEE 801.11p standard—to enable vehicles and intersections to communicate over distances of 300 meters or more [Denso Corporation, 2006]. These WSUs are no more expensive than a standard traffic-signal installation (on the order of a few thousand dollars) and can be deployed quickly and easily. This same hardware could fill the hardware require-

ments for driver agents as well—it is engineered to withstand the extreme conditions of operation on a motor vehicle. The marginal cost per-vehicle to add such equipment would likely be even lower, as most vehicles already contain sophisticated computer hardware that could handle much of these responsibilities. Economies of scale would further reduce the costs of such hardware. The reservation paradigm also relies on all agents having reasonably synchronized clocks. Fortunately, such synchronization can be easily achieved using any combination of GPS and network-based clock synchronization systems like the Network Time Protocol (NTP).

2.2.3 V2V

At intersections with very light to moderate traffic, the cost of installing, running, and maintaining the hardware required to operate an intersection manager may outweigh the benefits. In cases such as this, it would be nice still to get some of the benefits of autonomous vehicles. In particular, if vehicles can cross without stopping and with little slowing when few other vehicles are present, the intersection will still be much more efficient than comparable modern-day intersections. In order to attain this efficiency, the vehicles must resolve conflicts without the assistance of an arbiter agent. While this problem is well-understood in distributed systems, the added complexity of moving vehicles, each with substantial inertia, makes this case more interesting in this context. This scenario is referred to as a "Vehicle-to-Vehicle" (V2V) scenario, as vehicles communicate only with each other.

Distributed Consensus

In distributed systems, the problem of *consensus* is that of getting a group of processes to agree in the presence of potential failures [Pease *et al.*, 1980]. While this problem can be impossible to solve in bounded time [Fischer *et al.*, 1985], the problem solved in the V2V scenario differs in two important ways. First, we insist that

all messages sent between vehicles are digitally signed to ensure that the identity of the sending vehicle is known. If the identity of the sending vehicle can not be ascertained with absolute certainty, the message is ignored. Vehicles can thus be held responsible for their messages. We set up the algorithm in such a way that, with some minor external monitoring (similar to modern-day speed-traps), driver agents have no incentive to falsify their information. Second, we do not technically need a full consensus every time—we only need to make sure that vehicles only enter the intersection if they will not collide with one another. If for some extremely unlikely reason the vehicles cannot agree, none of them will enter the intersection, and the system will fall back to a stop sign–like mode.

The idea behind our V2V paradigm is that vehicles broadcast intended trajectories to all nearby vehicles, and via a conflict-resolution algorithm, come to an agreement on which vehicles are permitted to follow their broadcasted trajectories and which are not. While such a paradigm can lead to vehicle collisions in theory, as a result of massive (and unnoticed) communication failures, a production system can be built to guarantee that communication failures of this sort do not occur.

Technological Basis

In the absence of a fixed transmitter and receiver, the V2V scenario requires vehicles to communicate over longer distances than a comparable V2I scenario. Instead of communicating with an agent stationed at the intersection, a vehicle 200 meters from the intersection may need to communicate with a different vehicle 200 meters from the intersection in the opposite direction. Most of the time it will not be an issue, as these intersections are specifically chosen to have very little traffic. Thus, if a conflict exists between two vehicles' exepected trajectories, there should be ample space-time in the intersection to accomodate them both with only a small adjustment. However, even intersections with light traffic sometimes experience unexpected temporary

bursts of traffic. In this case, an ad-hoc wireless dissemination algorithm like RAPID can be used to ensure that the information in vehicles' transmissions reaches even the furthest approaching vehicles [Drabkin *et al.*, 2007]. Given that current automotive wireless communications standards include reliable transmission at ranges of 300 meters and greater, and that these distances will continue to grow as the technology improves, such sophisticated solutions will likely be unnecessary. As with the V2I scenario, the V2V system relies on synchronized clocks. Since we again assume that vehicles have access to GPS signals, sufficiently accurate synchronization is readily available.

Chapter 3

Communication Protocol

The key difference between a multiagent system and a system that just happens to have multiple agents is the interactions of the agents. In order for agents to interact, there must be some standardized mechanism for communication. Whether this communication is *stigmergic*, such as in swarm-intelligences like a colony of ants or bees, or more direct and intentional, some mechanism must exist. By creating an explicit communication protocol, we define the system for which a limitless variety of agents can be created. In this chapter, I present our communication protocol, which uses both direct agent-to-agent and broadcast transmissions. The protocol consists of a fixed set of message types, each with various fields for storing information, as well as rules that must be obeyed concerning the sending and receiving of these messages, as well as the actions that may or may not be taken by an agent that has received or sent them. Some of these message types, fields, and rules (marked with a †) are additions necessary for networks of multiple intersections. These messages often refer to the Admission Control Zone (ACZ), a region beyond the intersection to which the intersection manager can control access. The ACZ is discussed in detail in Chapter 10.

3.1 Message Types

The vehicles and intersection manager are each restricted to a few types of messages with which they must coordinate.

3.1.1 Vehicle → Intersection

There are eight types of messages that can be sent from vehicles to the intersection.

Request

A driver agent sends the REQUEST message in two cases: either it does not have a reservation and wishes to make one, or it has a reservation and wishes to change it. Each REQUEST can contain multiple *proposals*, ordered from most desirable to least desirable. In addition to one or more proposals, the REQUEST contains the properties of the vehicle (ID number, size, etc.). Each proposal contains the properties of the proposed reservation (arrival time, arrival velocity, arrival lane, etc.). This message also communicates the vehicle's status as an emergency vehicle (in an emergency situation). In practice, this would be implemented using a secure method such that normal vehicles could not impersonate emergency vehicles. Such methods are well understood and the details of the implementation are beyond the scope of this research.

This message has 13 fields:

`source_id` — the vehicle's unique identification number (VIN).

`destination_id` — the identification number of the intersection manager to which the message should be delivered.

`vehicle_length` — the length of the vehicle in meters.

`vehicle_width` — the width of the vehicle in meters.

`maximum_acceleration` — the maximum rate, in meters per second squared, at which the vehicle can accelerate.

`minimum_acceleration` — the minimum rate, in meters per second squared, at which the vehicle can accelerate (i.e. negative number representing maximum deceleration).

`minimum_velocity` — the minimum velocity, in meters per second, at which the vehicle can travel (usually a negative value indicating travel in reverse).

`front_wheel_displacement` — the distance, in meters, between the front of the vehicle and the front axle.

`rear_wheel_displacement` — the distance, in meters, between the front of the vehicle and the rear axle.

`max_steering_angle` — the maximum number of radians in either direction away from directly ahead (0) to which the front wheels can be turned for the purposes of steering.

`max_turn_per_second` — the maximum angular velocity, in radians per second, at which the vehicle can turn its wheels.

`emergency` — a Boolean value representing whether or not this is an emergency vehicle in an emergency situation.

`traversal_proposals` — a list of proposals for traversing the intersection, ordered from highest to lowest desirability. Each proposal has 5 fields:

`arrival_lane` — a unique identifier for the lane in which the vehicle will be when it arrives at the intersection.

`departure_lane` — a unique identifier for the lane in which the vehicle desires to exit the intersection.

`arrival_time` — the absolute time at which the vehicle will arrive at the intersection.

`arrival_velocity` — the velocity, in meters per second, at which the vehicle expects to be traveling when it arrives at the intersection.

`maximum_velocity` — the maximum velocity, in meters per second, at which the vehicle can make the indicated traversal.

Cancel

A driver agent sends a CANCEL message when either it no longer desires its current reservation or it no longer believes it can maintain its current reservation.

It has 3 fields:

`source_id` — the vehicle's unique identification number (VIN).

`destination_id` — the identification number of the intersection manager to which the message should be delivered.

`reservation_id` — the unique identifier for the reservation to be canceled, which was obtained from the intersection when the reservation was confirmed.

Done

This message is sent when the vehicle has completed its traversal of the intersection.

It has 2 fields:

`source_id` — the vehicle's unique identification number (VIN).

`destination_id` — the identification number of the intersection manager to which the message should be delivered.

Away†

This message is sent when the vehicle has exited the Admission Control Zone (ACZ) for the lane in which it currently is. The ACZ is an area beyond the intersection to which the Intersection Manager can control access. See Chapter 10 for a full description.

It has 2 fields:

`source_id` — the vehicle's unique identification number (VIN).

`destination_id` — the identification number of the intersection manager to which the message should be delivered.

ACZRequest†

A driver agent sends an ACZRequest when it wants to enter a lane within an Admission Control Zone (ACZ).

It has 5 fields:

`source_id` — the vehicle's unique identification number (VIN).

`destination_id` — the identification number of the intersection manager to which the message should be delivered.

`start_lane` — a unique identifier for the lane in which the vehicle currently is, or -1 if the vehicle is not currently in a lane.

`target_lane` — a unique identifier for the lane the vehicle wishes to enter.

`vehicle_length` — the length of the vehicle in meters.

ACZCancel†

A driver agent sends an ACZCancel when it no longer wants to or no longer believes it is able to enter a lane in accordance with the parameters of the latest ACZConfirm it has received.

It has 3 fields:

`source_id` — the vehicle's unique identification number (VIN).

`destination_id` — the identification number of the intersection manager to which the message should be delivered.

`ticket_number` — the highest `ticket_number` of any ACZConfirm received by the driver agent from the current intersection manager, which represents the current agreement for the driver agent's vehicle to enter a lane within the Admission Control Zone (ACZ).

ACZEntered†

A driver agent sends an ACZEntered when it has finished entering a lane within an Admission Control Zone (ACZ) via the parameters of the latest ACZConfirm it has received.

It has three fields:

`source_id` — the vehicle's unique identification number (VIN).

`destination_id` — the identification number of the intersection manager to which the message should be delivered.

`ticket_number` — the `ticket_number` of the ACZConfirm in accordance with which the driver agent changed lanes.

ACZExit†

A driver agent sends an ACZExit when it leaves an Admission Control Zone (ACZ) by exiting the roadway, as opposed to by getting far enough away from the intersection.

It has two fields:

`source_id` — the vehicle's unique identification number (VIN).

`destination_id` — the identification number of the intersection manager to which the message should be delivered.

3.1.2 Intersection → Vehicle

There are five types of messages that can be sent from intersections to vehicles.

Confirm

This message is a response to a vehicle's Request message. It does not always mean that one of the proposals transmitted by the vehicle were acceptable. It could, for example, contain a counter-offer by the intersection. The reservation parameters in this message are implicitly accepted by the vehicle, and must be explicitly canceled if the driver agent of the vehicle does not approve. Note that this "push" semantics is safe even with faulty communication—the worst that can happen is that the intersection reserves space that does not get used. Included in the message are acceleration constraints determined by the intersection. These are stored as a run-length encoded list of rates and durations. How the list is created depends on the intersection manager. The vehicle's safety must be guaranteed if it adheres to the list.

It has 11 fields:

`source_id` — the identification number of the intersection manager sending the message.

`destination_id` — the unique identification number (VIN) of the vehicle to which the message should be delivered.

`reservation_id` — a unique identifier for the reservation just created. This value increases monotonically to ensure that the recipient knows which message is the most recent.

`arrival_time` — the absolute time at which the vehicle is expected to arrive.

`early_error` — the maximum amount of time, in seconds, before the `arrival_time` at which the vehicle may arrive at the intersection.

`late_error` — the maximum amount of time, in seconds, after the `arrival_time` at which the vehicle may arrive at the intersection.

`arrival_lane` — a unique identifier for the lane in which the vehicle should arrive at the intersection.

`departure_lane` — a unique identifier for the lane in which the vehicle should depart the intersection.

`arrival_velocity` — the velocity, in meters per second, at which the vehicle should arrive at the intersection. A negative number signifies that any velocity is acceptable.

`acz_distance`† — the length of road, in meters, after the intersection, that is governed by an Admission Control Zone (ACZ).

`accelerations` — a run-length encoded description of the expected acceleration, in meters per second squared, of the vehicle as it travels through the intersection. Here, a run-length encoded description is a sequence of order pairs

of *acceleration* and *duration*—starting with the instant the vehicle enters the intersection, it should maintain each *acceleration* for the *duration* with which it is paired. If the sequence is empty, this signifies that any accelerations are acceptable, however vehicles are responsible for maintaining a safe distance from vehicles in front of them.

Reject

By sending a REJECT message, an intersection manager can inform a driver agent that none of the parameters sent in the latest REQUEST were acceptable, and that the intersection either could not or did not want to make a counter-offer.

This message has 4 fields:

`source_id` — the identification number of the intersection manager sending the message.

`destination_id` — the unique identification number (VIN) of the vehicle to which the message should be delivered.

`next_communication` — the absolute time at which subsequent REQUEST messages will be accepted by the intersection manager.

`reason` — the reason why the REQUEST was rejected. This can be one of 12 values:

- `MALFORMED` — the REQUEST message was not formed properly, or contained no traversal proposals.
- `EMERGENCY` — the intersection is currently shut down due to an emergency situation
- `TIMEOUT` — the REQUEST was sent before the sending vehicle was permitted to do so.

`EMERGENCY_VEHICLE` — the intersection manager is temporarily suspending reservations in all of the proposed arrival lanes due to the presence of an emergency vehicle.

`TURN_FORBIDDEN` — the proposed combination of arrival and departure lanes is not permitted by the intersection manager.

`TIME_TRAVEL` — the proposed `arrival_time`s are all in the past.

`FUTURE_LIMIT` — the proposed `arrival_time`s are all too far in the future.

`RESERVATION_DISTANCE` — the intersection manager is not currently granting reservations to vehicles arriving in any the proposed lanes that are as far from the intersection as the requesting vehicle.

`STOP_REQUIRED` — the intersection manager will not grant a reservation except in the case that the requesting vehicle is stopped at the intersection.

`NO_CLEAR_PATH` — the intersection manager was unable to find a clear, safe path through the intersection using any of the parameters provided.

`ACZ_CAPACITY`† — the intersection manager did not have sufficient capacity in the Admission Control Zone for a proposed departure lane.

`NONE` — no reason given.

Emergency-Stop

The Emergency-Stop message is sent when the intersection manager has determined that a collision or similar problem has occurred in the intersection. This message informs the receiving driver agent that no further reservation requests will be granted, and if possible, the vehicle should attempt to stop instead of entering the intersection, even if it has a reservation.

It has 2 fields:

`source_id` — the identification number of the intersection manager sending the message.

`destination_id` — the unique identification number (VIN) of the vehicle to which the message should be delivered.

ACZConfirm†

The ACZConfirm message is a response to a vehicle's ACZRequest indicating that the vehicle may change lanes inside an Admission Control Zone (ACZ).

It has 6 fields:

`source_id` — the identification number of the intersection manager sending the message.

`destination_id` — the unique identification number (VIN) of the vehicle to which the message should be delivered.

`ticket_number` — a unique identifier for this confirmation. This value increases monotonically to ensure that the recipient knows which message is the most recent.

`start_lane` — a unique identifier for the lane in which the vehicle has permission to change out of, or -1 if the vehicle is not currently in a lane.

`target_lane` — a unique identifier for the lane the vehicle has permission to enter.

`acz_distance` — the length of road, in meters, after the intersection, that is governed by an Admission Control Zone (ACZ).

ACZReject†

An ACZReject message is a response to a vehicle's ACZRequest indicating that the vehicle may not change lanes as requested.

It has 3 fields:

`source_id` — the identification number of the intersection manager sending the message.

`destination_id` — the unique identification number (VIN) of the vehicle to which the message should be delivered.

`reason` — the reason why the ACZRequest was rejected. This can be one of 4 values:

`MALFORMED` — the Request message was not formed properly, or contained no traversal proposals.

`EXISTING_TICKET` — the vehicle already has permission according to a previous ACZConfirm and must cancel first.

`ACZ_CAPACITY` — the intersection manager did not have sufficient capacity in the Admission Control Zone to permit the lane change.

`NONE` — no reason given.

3.1.3 Vehicle → Vehicle

In the event that no intersection manager is available, vehicles can coordinate using the vehicle-to-vehicle (V2V) messages. While not as efficient or robust as vehicle-to-intersection (V2I), this portion of the protocol can be used at intersections without any additional infrastructure. There are two types of messages that vehicles can send to one another. Much of the work on V2V scenarios was done in collaboration with Mark VanMiddlesworth, whom the author would like to recognize for his contribution.

Claim

The CLAIM message is broadcast by a vehicle when it wishes to stake out space-time in an intersection not governed by an intersection manager.

It has 8 fields:

`source_id` — the sending vehicle's unique identification number (VIN).

`message_id` — a monotonically increasing number greater than that of any different previously broadcast message.

`intersection_id` — a unique identifier for the intersection the vehicle wishes to traverse.

`stopped_at_intersection` — whether or not the vehicle is stopped at the intersection.

`arrival_lane` — a unique identifier for the lane in which the vehicle plans to arrive at the intersection.

`departure_lane` — a unique identifier for the lane in which the vehicle plans to depart the intersection.

`arrival_time` — the absolute time at which the vehicle plans to arrive at the intersection.

`departure_time` — the absolute time at which the vehicle plans to depart the intersection.

Clear

The CLEAR message is broadcast by a vehicle when it no longer wishes to traverse the intersection.

It has 3 fields:

`source_id` — the sending vehicle's unique identification number (VIN).

`message_id` — a monotonically increasing number greater than that of any different previously broadcast message.

`intersection_id` — a unique identifier for the intersection for which the vehicle no longer wishes to stake out its previous CLAIM.

3.2 Protocol Actions

Because the protocol contains both vehicle-to-intersection (V2I) and vehicle-to-vehicle (V2V) components, there are two distinct sets of rules that agents must follow, depending on which category describes the scenario. In the V2I case, there are rules for both the driver agents in the vehicles and the intersection managers stationed at the intersection. In the V2V case, as there is no intersection manager, there are only rules for the driver agents.

3.2.1 V2I Rules

In the V2I scenario, driver agents rely on the intersection manager to ensure that they cross the intersection safely. Because the system must be robust to communications failures, there is an inherent "push" semantics, meaning that when the intersection sends a CONFIRM (or ACZCONFIRM) message, the intended recipient agrees with stipulations in those messages implicitly. If, in fact, the driver agent does not agree, the burden is on that driver agent to communicate that fact. The rules and the messages are designed around this semantics to ensure that if a message is lost, the worst possible outcome is space-time reserved in the intersection that goes unused, and not a collision.

Driver Agent Rules

As part of the V2I protocol, driver agents controlling vehicles must obey the following rules:

1. A vehicle may only traverse an intersection in accordance with reservation parameters contained in a CONFIRM message sent by that intersection's intersection manager.

2. † A vehicle may only enter the roadway or change lanes inside the Admission Control Zone (ACZ) of an intersection in accordance with the parameters contained in a ACZCONFIRM message sent by the intersection manager controlling the intersection out of which the roadway or lanes exit.

3. A vehicle must transmit a DONE message when it has completed its traversal of the intersection.

4. † A vehicle must transmit an AWAY message when it has reached a point at least as far away from the intersection as the `acz_distance` parameter in the CONFIRM or ACZCONFIRM which permitted it to traverse the intersection or change lanes, respectively.

5. † If the `acz_distance` parameter in the CONFIRM or ACZCONFIRM message received by a vehicle is zero or less, the vehicle is itself solely responsible for determining whether it is safe to depart the intersection or change lanes.

6. † Vehicles must make every effort to clear an intersection's Admission Control Zone expeditiously.

Intersection Manager Rules

The intersection manager must make the following guarantee:

1. If a vehicle crosses the intersection in accordance with the most recent CONFIRM message issued by the intersection manager to that vehicle, it must be safe from collisions while in the intersection.

3.2.2 V2V Rules

The foundation upon which the V2V portion of the protocol rests is that driver agents have an understanding about which vehicle is expected to yield in any given situation. By having a mutually accepted and understood set of rules, any individual driver agent knows that violating them could result in a collision involving its vehicle. Just as a human driver knows to stop at a red signal (even though nothing prevents him or her from driving through), agents will necessarily be designed to follow the rules, lest their vehicles and occupants be damaged. Each vehicle, given complete information, will will run the same algorithms, and get the same results.

Conflict, Priority, Dominance, and Permissibility

To facilitate the description of the rules governing the V2V protocol, we define the following relations on CLAIM messages.

Conflict Two CLAIM messages are said to *conflict* if all of the following are true:

- The `intersection_id` fields of the two messages are identical.
- The paths determined by the `arrival_lane` and `departure_lane` fields are not *compatible* (compatible paths do not intersect).
- The time intervals are not disjoint.

Priority We define the relative *priority* of two CLAIM messages based on the following rules, presented in order from most significant to least significant:

1. If one message specifies an `arrival_time` in the past (i.e. the vehicle is already in the intersection) and another message specifies an `arrival_time` in the future, the message with the `arrival_time` in the past has priority.

2. If one message specifies that the sending vehicle is stopped at the intersection, but the other does not, the CLAIM that specifies the sending vehicle is stopped at the intersection has priority.

3. If neither message specifies that the sending vehicle is stopped at the intersection, the CLAIM with the earliest `exit_time` has priority.

4. If both messages specify that the sending vehicles are stopped at the intersection, and are in roads that are the same or duals of each other (the same road but in a different direction), then if one message has the same `arrival_lane` as its `departure_lane`, that message has priority.

5. If both messages specify that the respective sending vehicles are stopped at the intersection, the CLAIM whose `arrival_lane` is "on the right" has priority. Here, "on the right" is defined similarly to current traffic laws regarding four-way stop signs. This binary relation on the incident lanes is globally available as a characteristic of the intersection.

6. If priority cannot be established by the previous rules, the CLAIM with the lowest `vehicle_id` has priority.

.

Dominance Given two claims c_1 and c_2, we say that c_1 *dominates* c_2 if c_1 and c_2 conflict and c_1 has priority over c_2. The *dominance graph* G of a set of claims $C = \{c_1, c_2, \ldots, c_n\}$ is a digraph with vertices $V(G) = \{v_1, v_2, \ldots, v_n\}$, and directed edges $E(G) = \{(v_i, v_j) | c_i \text{ dominates } c_j\}$.

Permissibility Each CLAIM in a set of CLAIMS C is classified as either *permissible* or *nonpermissible* by repeating the following steps in order until all CLAIMS are classified.

1. For each $c_i \in C$, if $\forall c_j \in C$ such that c_j dominates c_i and c_j is classified nonpermissible, c_i is classified permissible.

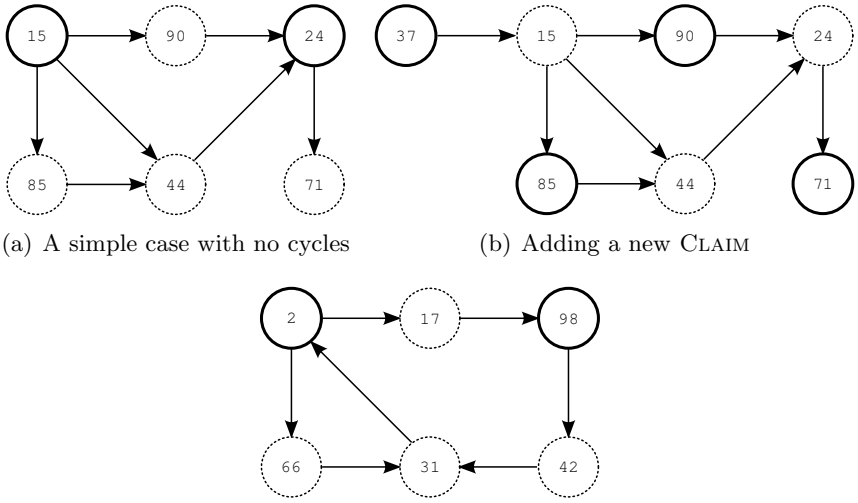
2. For each $c_i \in C$, if $\exists c_j \in C$ such that c_j dominates c_i or c_i dominates c_j, and c_j is classified permissible, c_i is classified nonpermissible.

3. Let $\mathcal{C} \subset C$ be the current set of unclassified CLAIMS. Let $\mathcal{C}_{\text{dom}}$ be $\{c \in \mathcal{C} | \exists d \in \mathcal{C}, c \text{ dominates } d\}$. Let $c_{\min} \in \mathcal{C}_{\text{dom}}$ be the CLAIM in $\mathcal{C}_{\text{dom}}$ with the lowest VIN. Classify $c_{\min}$ permissible. Perform this step only if no vertices were labeled in steps 1 or 2. In practice, this is an exceedingly remote possibility.

This protocol is designed with the goal of incentive compatibility: reducing opportunities for vehicles to benefit by misrepresenting their traversal parameters. We thus consider two important theoretical properties of this classification of CLAIMS. First, it is a partition: all CLAIMS are either permissible or nonpermissible. Second—and most importantly—the set of permissible CLAIMS in C corresponds to a maximal independent set in the dominance graph of C (adding any additional CLAIM from C would remove its independence). If any vehicle with a nonpermissible CLAIM attempts to traverse the intersection, it does so at its own peril. Note that in general, finding a *maximum* independent set is NP-complete, and therefore the algorithm may not always return the largest possible set of CLAIMS [Garey and Johnson, 1979].

Figure 3.1 shows three dominance graphs, in which vertices are labeled with the VIN of the CLAIM with which they correspond. In 3.1(b), the CLAIM with VIN 37 is added, substantially altering the partition of permissible and nonpermissible CLAIMS. In 3.1(c), a cycle is broken by making the CLAIM with VIN 2 permissible,

however, this does not yield the maximum independent set, which would be the Claims with VINs 17, 42, and 66.



(a) A simple case with no cycles

(b) Adding a new Claim

(c) Breaking cycles requires step 3

Figure 3.1: Three different dominance graphs for three sets of Claims. A directed edge indicates dominance, a bold circle indicates a permissible Claim, while a dashed circle indicates a nonpermissible Claim.

Driver Agent Rules

A driver agent using the V2V protocol to traverse an intersection must obey the following rules:

1. A vehicle may not enter the intersection if its current Claim is nonpermissible in the set of current Claims of which it is aware. Because the permissible Claims form a maximal independent set, any vehicle that violates this rule risks a collision with another vehicle.

2. A vehicle may not enter the intersection without first broadcasting its Claim for at least 0.5 seconds.

3. A vehicle must vacate (or make every effort to vacate) the intersection at or before the `exit_time` specified in its most recent CLAIM.

4. A vehicle must follow a reasonable path from the point of entry into the intersection to the point of departure out of the intersection. This means, for example, that a vehicle going straight through the intersection without turning must remain within its lane, while a vehicle turning must not interfere with any compatible paths through the intersection.

5. The `stopped_at_intersection` field of an agent's CLAIM must be `true` if and only if the agent's vehicle is stopped at the intersection.

6. The `arrival_time` field of an agent's CLAIM must be in the past if and only if the agent's vehicle is in the intersection.

Effects of Communication Failure

Because the V2V protocol does not have the "push" semantics of the V2I protocol, communication failures can cause safety failures. Because each driver agent must construct its own dominance graph, if messages are lost, the graph may be altogether different. As Figure 3.1 demonstrates, a single vertex in the graph can change which CLAIMS are permissible. This situation could, in turn, lead to a collision if a driver agent mistakenly determines that its claim is permissible. Because each message is repeatedly broadcast, a vehicle would have to fail to receive every message from a vehicle not to notice its presence. Such a scenario is possible in the presence of complete communication failure on the part of the sending or receiving vehicle. Such a failure should be easy to detect with redundant transmitters and receivers, as well as self-diagnostics.

In order to disseminate each vehicle's CLAIM as quickly and reliably as possible, more sophisticated communication mechanisms can be used, which are specif-

ically designed for the challenges of ad-hoc wireless networks [Drabkin *et al.*, 2007]. In these mechanisms, message headers are rebroadcast by all vehicles, such that a vehicle can detect if it has missed a message. In the case that it has, it can request the full message from the vehicle that rebroadcast the header. For example, imagine three vehicles, A, B, and C. Vehicle C cannot send or receive messages to or from vehicle A, but all other communication channels are functioning. In this case, when vehicle B receives vehicle A's CLAIM, it will include that CLAIM's header the next time it rebroadcasts its own CLAIM. Vehicle C will then notice it has missed this message, and can request that vehicle B retransmit the message in its entirety on behalf of vehicle A. The same will be true for any messages of vehicle C's that vehicle A missed. Considerations for reliable and quick dissemination are extremely important in the V2V scenario. If a single vehicle fails to detect even a single other vehicle, the system has violated its safety properties, even if no collision ensues.

Chapter 4

Intersection Manager Implementation

While a protocol can specify the ways that agents can interact in a multiagent system, it does not specify how those agents will act individually, or which of the interaction options they will select. As long as an agent obeys the protocol, it can take any actions it desires. The protocol described in Chapter 3 is meant to support many heterogenous implementations simultaneously. In order to evaluate the performance capabilities of the protocols, we must create an implementation for each type of agent. This thesis demonstrates what is possible with the protocol using just one implementation. In this chapter, I present our implementation of the intersection manager, one of which is stationed at each intersection. First, I introduce FCFS, the intersection control policy we use for the majority of our experiments. Next, I present some heuristics that we use to make the work of the intersection control policy easier. Last, I enumerate several alternate policies that we compare with FCFS in experiments.

4.1 The FCFS Policy

Although the intersection manager communicates directly with the driver agents, the intersection control policy is the "brains" behind the operation. Recall that vehicles need not know anything about the policy, just the protocol. The policy implementation is completely private to the intersection manager. In this section, I describe the main intersection control policy used in this work. Because of the "First Come, First Served" nature of the policy, we call policy "FCFS". The main part of the policy—the request processing—is shown in Algorithm 1.

FCFS enables a car to reserve in advance the space-time it needs to cross the intersection. Planning ahead allows vehicles coming from all directions to traverse the intersection simultaneously with minimal delay. The policy works as follows:

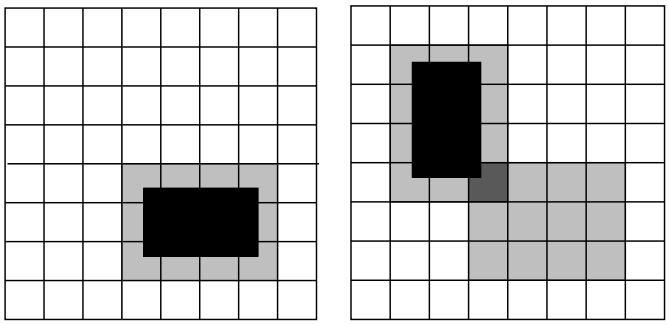
- The intersection is divided into a grid of square *reservation tiles*, each of which measures g meters on a side. We call g the *granularity* of the policy. The *granularity ratio* of the policy is defined as $\frac{\sqrt{A}}{g}$, where A is the total area of the intersection. An intersection that consists of exactly one reservation tile would thus have a granularity ratio of 1, while a square intersection made up of four reservation tiles would have a granularity ratio of 2. A reservation tile is essentially a map from times to vehicle ID numbers. If the map contains the $(key, value)$ pair (t, v), then vehicle v has the tile reserved at time t. Because our simulator is discretized, our implementation discretizes the map. However, reservation tiles could also be implemented by mapping intervals instead of discrete times.

- Upon receiving the reservation parameters from an approaching driver agent, the policy runs two *internal simulations* of the trajectory of the vehicle across the intersection using these parameters (line 6). The first simulation allows the vehicle to accelerate while in the intersection (line 27), whereas the second

simulation does not. The simulation lasts until the vehicle exits the intersection (line 10).

- At each time step of the internal simulation, the policy determines which reservation tiles will be occupied by the vehicle (line 11).
- If at any time during the simulation the requesting vehicle occupies a reservation tile that is already reserved by another vehicle (line 19), the policy rejects the request (line 23). Otherwise, the policy accepts the reservation and reserves the appropriate tiles for the times they will be required (line 35).

Figure 4.1 shows a graphical depiction of the concept behind the FCFS policy.



(a) Successful (b) Rejected

Figure 4.1: The internal simulation of an FCFS policy. The black rectangles represent vehicles, and the shaded tiles are tiles that are currently reserved. In 4.1(a), a vehicle's request is accepted, and the intersection reserves a set of tiles at time t. In 4.1(b), a second vehicle's request is rejected because during the simulation of its trajectory, the policy determines that it requires a tile (darkly shaded) already reserved by the first vehicle at time t.

While the concept behind FCFS is sound, it requires some modifications before it will work reliably, safely, and efficiently—even in simulation. In the remainder of this section, I present these modifications, namely buffers and edge tiles, which were created in response to early experimental results documented in Chapter 7. Videos of the `aim2` simulator running the FCFS policy can be seen on the videos

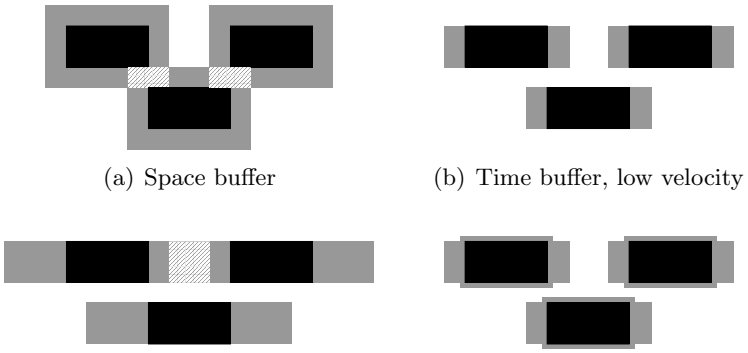
section of the project page at `http://www.cs.utexas.edu/~kdresner/aim/`.

4.1.1 Buffers

In any system involving physical robots, noise in sensor readings and errors in actuators will inevitably manifest themselves. Even in simulation, artifacts resulting from the discretization of time are enough to weaken the reservation tiles' guarantees of exclusivity. In the intersection, where vehicles move at high speeds in all different directions, these potential sources of calamity cannot be ignored. For example, what happens when a driver agent realizes that it will not make its reservation exactly on time, close enough to the intersection that it is not possible to stop before entering the intersection? Some sort of safety buffer is required. Two types of buffers are most natural: space buffers and time buffers. These buffers can be adjusted to guard against arbitrarily large sensor errors, at an efficiency cost.

Space buffers—buffers whose size is constant—suffice for safety purposes. If the intersection manager assumes each vehicle is ten times as large in each dimension, no vehicle should even get close to another vehicle. However, this excessive caution defeats the point of the intersection manager, which is to leverage the increased precision of autonomous vehicles. Furthermore, a space buffer does not take into account the direction of motion of the vehicle. Two vehicles whose paths would never intersect may begin to interfere with one another's reservation process if a large space buffer is used, as in Figure 4.2(a).

Time buffers, on the other hand, do take into account the motion of the vehicles. If the intersection manager instead assumes that the vehicle might be early or late, the actual area restricted by this buffer will shrink and grow with the vehicle's velocity, and only in the direction of movement. Figures 4.2(b) and 4.2(c) show how the buffer scales with the speed of the vehicle. Thus, if two vehicles are traveling along parallel lines, the time buffers for those vehicles should not interfere



(a) Space buffer (b) Time buffer, low velocity

(c) Time buffer, high velocity (d) Hybrid buffer

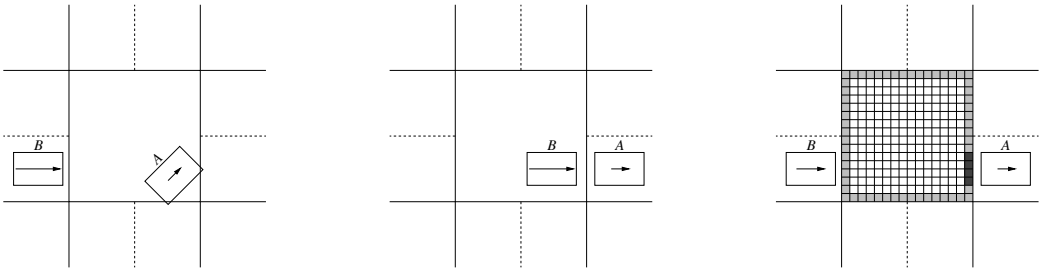
Figure 4.2: Various styles of buffers designed to cope with sensor noise and actuator errors. The hatched areas show where buffers would cause reservation conflicts: only one of each pair of conflicting vehicles would be granted a reservation.

unless those vehicles could potentially collide (they are in the same lane or the lanes are too close together for the vehicles' width). Alone, time buffers are not sufficient to guarantee safety — a small error in lateral positioning (orthogonal to the direction of motion) may still cause a collision. Figure 4.2(d) shows the best solution: a *hybrid buffer*. The hybrid buffer has a time buffer that scales with velocity, as well as a small space buffer that protects against lateral positioning errors and serves as a minimum buffer for slow-moving vehicles.

4.1.2 Edge Tiles

When driving on the open road, vehicles must maintain a reasonable following interval (usually measured as an amount of time) between one another. If a vehicle decelerates suddenly, it puts the vehicle behind it in a dangerous situation—if the rear vehicle doesn't react quickly enough, it may collide with the front vehicle. In the intersection, following intervals are not very practical, because vehicles are traveling in many different directions. Vehicles in the intersection cannot react normally to their sensor readings, because the intersection manager may orchestrate some "close

calls" that would look like a potential collision to a vehicle operating in "open road" mode. Instead, the vehicles trust the constraints given to them by the intersection manager. The intersection can guarantee this is safe in the intersection, but when a vehicle exits the intersection, it may enounter a vehicle that also just left the intersection, but at a much slower velocity. As shown in Figures 4.3(a) and 4.3(b), this situation may lead to an unavoidable collision, with the later vehicle being unable to stop quickly enough. Even with autonomous vehicles, which can react almost instantaneously, some amount of following interval is required for vehicles leaving the intersection.



(a) A turns right in front of B.

(b) B cannot stop in time.

(c) B must slow down preemptively.

Figure 4.3: Edge tiles prevent collisions after vehicles leave the intersection. In 4.3(a), vehicle A turns in front of vehicle B, traveling slowly because it is making a right turn. In 4.3(b), vehicle B gets through the intersection without incident, but finds that once it leaves the intersection, it cannot stop before colliding with vehicle A. The extra buffers on edge tiles, as shown in 4.3(c), prevent vehicle B from obtaining a reservation which would cause it to exit the intersection too close to vehicle A. The shaded tiles are edge tiles, while the darkly shaded tiles are the specific tiles that would prevent the collision in 4.3(a) and 4.3(b).

A first-cut solution to this problem is simply to increase the time buffers on all reservation tiles to the desired following interval. Thus, if vehicles require a following interval of one second when exiting the intersection, then no vehicle will be able to reserve a tile within one second of another vehicle. Vehicles leaving the intersection in the same lane will not exit within one second of each other, and there

will be a gap of at least one second between the vehicles. Unfortunately, this wreaks havoc with FCFS's ability to conduct vehicles efficiently through the intersection. The "close calls" from which the system gets its efficiency advantages will no longer be possible.

Instead, we divide the reservation tiles into two groups. *Internal tiles* are tiles that are surrounded on all sides by other reservation tiles. *Edge tiles*, which are shown shaded in Figure 4.3(c), are tiles that abut the intersection. At sufficiently high granularities, edge tiles are a relatively small fraction of the total number of tiles. It is only on these tiles that we increase the time buffer to the desired following interval. Because (at sufficiently high granularities) only vehicles leaving by the same lane will require the same edge tiles, this modification enforces the desired following intervals without otherwise preventing the intersection from exploiting its ability to interleave vehicles closely.

4.1.3 Emergency Vehicle Priority

In current traffic laws there are special procedures involving emergency vehicles such as ambulances, fire trucks, and police cars. Vehicles are required to pull over to the side of the road and come to a complete stop until the emergency vehicle has passed. This law exists both because the emergency vehicle may be traveling quickly, posing a danger to other vehicles, and because the emergency vehicle must arrive at its destination as quickly as possible—lives may be at stake. Once a system such as this is implemented, automobile accidents—a major reason emergency vehicles are dispatched—should be all but eradicated. Nonetheless, emergency vehicles will still be required from time to time as fires, heart attacks, and other emergencies will still exist. While we have previously proposed other methods for giving priority to emergency vehicles [Dresner and Stone, 2006], here we present a new, simpler method, which is fully implemented and tested.

Algorithm 1 FCFS's request processing algorithm. FCFS has persistent state variables: *tiles*, a map from tiles and times to vehicles, *reservations*, a map from vehicles to sets of tiles, and *timeouts*, a map from vehicles to times.

1: $t_c \leftarrow$ the current time
2: **if** *timeouts*[vehicle id] $< t_c$ **then**
3: reject the request
4: $t_a \leftarrow$ proposed arrival time
5: *timeouts*[vehicle id] $\leftarrow t_c + min(0.5, (t_a - t_c)/2)$
6: **for** *acceleration* in {true, false} **do**
7: *tile_times* $\leftarrow \{\}$
8: $t \leftarrow t_a$
9: $V \leftarrow$ temporary vehicle initialized according to reservation parameters
10: **while** V is in the intersection **do**
11: $S \leftarrow$ tiles occupied by V and V's space buffer at time t
12: *tile_times* $\leftarrow$ *tile_times* $\cup \{(t, S)\}$
13: **for all** $s \in S$ **do**
14: **if** s is an edge tile **then**
15: $buf \leftarrow$ edge tile buffer
16: **else**
17: $buf \leftarrow$ internal tile buffer
18: **for** $i = -buf$ to buf **do**
19: **if** *tiles*$[s, t + i]$ is reserved by another vehicle **then**
20: **if** *acceleration* **then**
21: goto line 29
22: **else**
23: reject the request
24: $t \leftarrow t +$ time step
25: move V according to physical model
26: **if** *acceleration* **then**
27: increase V's velocity by V's maximum acceleration
28: break
29:
30: **if** request is a change **then**
31: *old_tile_times* $\leftarrow$ *reservations*[vehicle id]
32: **for all** $(t_i, S_i) \in$ *old_tile_times* **do**
33: **for all** $s \in S_i$ **do**
34: clear reserved status of *tiles*$[s, t_i]$
35: **for all** $(t_i, S_i) \in$ *tile_times* **do**
36: **for all** $s \in S_i$ **do**
37: *tiles*$[s, t_i] \leftarrow$ vehicle id
38: *reservations*[vehicle id] $\leftarrow$ *tile_times*
39: accept request, return reservation constraints (incl. accelerations)

Augmenting The Protocol

In order to accommodate emergency vehicles, the intersection manager must first be able to detect their presence. The easiest way to inform the intersection manager is by including this information in a field of all request messages. In our protocol, this field is simply a flag that indicates to the intersection manager that the requesting vehicle is an emergency vehicle in an emergency situation (lights flashing and siren blaring). In practice, however, safeguards would need to be incorporated to prevent normal vehicles from abusing this feature in order to obtain preferential treatment. These safeguards could be implemented via a secret key instead of simply a boolean value, or even some sort of public/private key challenge/response mechanism. These details of the implementation, however, are beyond the scope of this project and are already a well-studied area of cryptography and computer security.

The FCFS-Emerg Policy

Now that the intersection control policy can detect emergency vehicles, it can process reservation requests while giving priority to the emergency vehicles. A first-cut solution is simply to deny reservations to any vehicles that are not emergency vehicles. However, this solution is not satisfactory, because if all the traffic comes to a stop due to rejected reservation requests, any emergency vehicles may get stuck in the resulting congestion. Instead, the FCFS-Emerg policy keeps track of which lanes currently contain approaching emergency vehicles. As long as at least one emergency vehicle is approaching the intersection, the policy grants reservations only to vehicles in those lanes. This restriction ensures that vehicles in front of the emergency vehicles will also receive priority. Due to this increase in priority, lanes with emergency vehicles tend to empty very rapidly, allowing emergency vehicles to proceed relatively unhindered.

To modify the algorithm shown in Algorithm 1, we add a new variable,

emergency_vehicles, which is a map from lanes to sets of vehicle IDs. If the map contains the $(key, value)$ pair $(l, [v_1, v_2])$, then emergency vehicles v_1 and v_2 are approaching in lane l. When a request comes in from an emergency vehicle, the proper mapping is added to *emergency_vehicles*. When a request comes in from a non-emergency vehicle in lane l, if *emergency_vehicles* contains any mappings (k, v) such that v is not an empty set, the request is rejected immediately unless *emergency_vehicles* contains a mapping (l, u) such that u is not empty. A video of the `aim2` simulator running the FCFS-EMERG policy can be found in the video section of the project page at `http://www.cs.utexas.edu/~kdresner/aim/`.

4.1.4 Safety Guarantees

Outside of the intersection, it is impossible to guarantee safety. All vehicles retain autonomy outside the intersection, and can thus do whatever they want, including deliberately crashing into other vehicles. However, with a few assumptions, we can make a guarantee about vehicles inside the intersection. Given that vehicles obey the protocol, and that they do not move extremely fast—for example, in such a way as to overlap so briefly that FCFS's internal simulation does not detect it with discretized time—we can guarantee that two vehicles will never occupy the same space at the same time. This guarantee follows quite trivially from the fact that each tile can be beneath only one vehicle at any time.

Outside of the intersection, we may not be able to guarantee safety, but we can make some safety-related guarantees. Again making the same assumptions about the way vehicles move, the edge tile buffers guarantee a certain following distance for all vehicles leaving the intersection. If the edge tiles have a time buffer of two seconds, we can know for sure that no two vehicles will exit the intersection in the same lane within two seconds of one another.

4.2 Other Policies

Because of the layer of abstraction provided by the protocol, the intersection manager can work in an emulation mode, imitating modern-day control mechanisms, such as the stop sign and traffic signal. Here we briefly explain the implementation of two intersection control policies designed to mimic these mechanisms.

Stop-Sign Stop signs are traditionally used at intersections with very light traffic. While they are much more cost-effective and reliable, they cannot provide the throughput and efficiency of a traffic signal. Thus, there would never be a reason for our system to emulate a stop sign, however we include a description for completeness.

STOP-SIGN is exactly like FCFS, except that it only accepts reservations from vehicles that are stopped at the intersection. Any other reservation requests are rejected with a message indicating the vehicle must stop at the intersection. The intersection determines whether a vehicle is stopped at the intersection by examining the difference between the current time and the arrival time in the request message.

Traffic-Light When the TRAFFIC-LIGHT policy receives a reservation request message, it calculates the next time after the proposed arrival time that a simulated (or real) signal for the sending vehicle's lane would be green. It then responds with a confirmation message that reflects this information. Because confirmation messages have maximum tolerable errors associated with them, the intersection manager uses these errors to encode the beginning and end of the green signal period. The TRAFFIC-LIGHT policy can actually be coordinated with real signal timings in order to allow an autonomous vehicle to use an intersection governed by a traffic signal without having to visually read the state of the signals.

4.3 Policy Switching

Because different policies perform differently under various traffic conditions, it would be useful to have an intersection manager that can switch policies, without having to bring the whole system to a halt. In this section, I explain how our intersection manager can switch smoothly between policies and briefly discuss the implications of this ability.

4.3.1 Smoothly Switching Between Two Policies

The simplest way for an intersection manager to switch between two intersection control policies is to refuse all reservation requests until the intersection is empty, at which point the manager could resume with the new intersection control policy. This naïve approach ignores the ability of the vehicles and intersection manager to plan ahead and schedule around the switchover. Our more efficient solution places only a small additional requirement on intersection control policies: each policy P must keep track of the latest time $last_P$ for which any vehicle could be in the intersection. Initially, $last_P$ is the current time, and P ensures that $last_P$ is always at least as late as the current time. Whenever P allows a vehicle through the intersection, it updates $last_P$ such that it is later than any time at which that vehicle could be in the intersection.

The vast majority of the time, the intersection manager will only be using a single policy. However, our intersection manager maintains a queue of policies $P_0, P_1, \ldots, P_n$, where $\forall 0 \leq i < j \leq n, last_{P_i} \leq last_{P_j}$. Whenever the intersection manager wants to switch to a new policy P_{n+1}, it *freezes* P_n, and enqueues P_{n+1}. If policy P is frozen, it rejects any reservation requests that would cause it to modify $last_P$. When a REQUEST message arrives, possibly containing multiple traversal proposals, the intersection manager must then decide which policy or policies to use to handle each of the traversal proposals. If a traversal proposal has an

`arrival_time` of t, it can only be handled by P_i, where i is the smallest value such that $t < last_{P_i}$. In other words, each policy P_i can only handle proposals that have the vehicle arrive and depart within the time interval $(last_{P_{i-1}}, last_{P_i}]$. Figure 4.4 provides a depiction of this process. Any proposal that would have a vehicle arrive within one policy's interval and depart in another cannot be accepted. Recall from Chapter 3.1.1 that the traversal proposals are ordered by priority. The first proposal is the one most desired by the vehicle, whereas the last is the least desired. The intersection manager should thus try to satisfy the proposals in order from first to last. Since each of these proposals may need to be handled by a different policy, each of which may have a different timeframe for responding, it is extremely complicated to develop a system that can test all of the proposals while preserving the preferences of the vehicle and ensuring that no vehicle holds more than one reservation at a time. Furthermore, situations in which this ability would be necessary are exceedingly remote. For that reason, we determine which policy should respond to the first traversal proposal, and then send that policy all the proposals within that policy's purview. All other proposals are ignored. This strikes a reasonable balance between complexity of implementation and catering to the preferences of the requesting driver agent.

4.4 Timeout

Once a driver agent's reservation request is rejected, that driver agent may immediately make a new request. Unless the new request is significantly different, it will most likely be rejected as well. With the exception of the request made immediately after the first rejected request, a driver agent's estimate of its arrival at the intersection is not likely to change much in the instant between consecutive requests. Eventually, after the vehicle has decelerated enough or the driver agents with conflicting reservations have canceled, the vehicle will obtain a reservation and make it

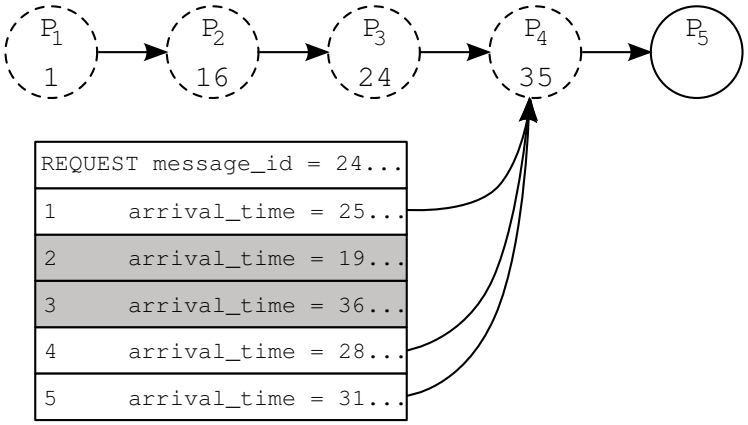


Figure 4.4: Five policies, P_1, P_2, ..., P_5 are all enqueued. The first four are frozen, and for each P_i, $last_{P_i}$ is shown below the policy name. A REQUEST message comes in with five traversal proposals. The first must be assigned to P_4 because the arrival time is too late for P_3. The second and third traversal proposals are ignored because they cannot be assigned to P_4, but the fourth and fifth proposals can. Note that any proposal that would cause the vehicle to leave the intersection after time $t > 35$ will be rejected, because $last_{P_4} = 35$.

through the intersection. From the standpoint of the intersection manager, each of the requests before the successful one are wasted effort. While our policy runs at most two internal simulations per request, those simulations may be computationally expensive, especially if the FCFS policy has a high granularity. Furthermore, if each rejected vehicle makes a request at every possible instant, the work can add up very quickly.

In order to keep the required amount of computation down and discourage driver agents from overloading the intersection manager with requests, the policy employs a system of *timeouts*. Once a driver agent's request is rejected, subsequent requests will not be considered until a period of time (determined by the reservation parameters) has elapsed. When rejecting a request, the policy includes in the rejection message the time after which it will consider further requests from the driver agent. In our implementation, this time is equal to $t + min(\frac{1}{2}, \frac{(t_a - t)}{2})$, where t is the

current time and t_a is the time of arrival in the request message. This process serves two purposes. First, it dramatically reduces the amount of computation the policy needs to do, because the intersection manager receives fewer requests. Vehicles may not obtain reservations at the earliest moment possible, but the computational savings are more than worth it. Second, it gives preference to vehicles that will enter the intersection sooner. If a vehicle is stopped at the intersection, it can send requests as quickly as it wishes, giving it the best chance of getting a reservation approved. A vehicle farther away, however, may have to wait the full half-second before attempting to make another reservation. As a vehicle approaches the intersection, if it is unable to procure a reservation, the frequency of opportunities to send reservation requests increases. In practice, timeouts significantly improve the performance of the system, allowing it to handle much higher traffic loads while avoiding backups.

4.5 Reservation Distance

Allowing accelerations in the intersection helps eliminate deadlocks, but other problems arose in our prototype implementation that significantly impaired the performance of the system. Frequently, a lane of traffic would become congested when many vehicles were spawned in that lane. Even when the simulator stopped spawning vehicles in that lane, the lane would remain congested. The problem is that FCFS, as first described, does nothing to control how vehicles in the same lane are alloted reservations. At best, the frontmost vehicle will get a reservation and make it through the intersection unhindered. However, this is often not the case. Sometimes the vehicle in front cannot obtain a reservation (due to congestion), and must decelerate. As shown in Figure 4.5, driver agents in vehicles further back may expect to accelerate soon and successfully reserve space-time in the intersection that the frontmost vehicle needs. While all vehicles will *eventually* make it through (a vehi-

cle might get a reservation immediately after vehicles behind it cancel), this process can repeat many times before the frontmost vehicle gets a reservation. In the worst scenarios, a single vehicle can continue for quite some time to obtain reservations that prevent the front car from crossing the intersection.

If we could maintain the invariant that vehicles do not get reservations unless all cars in front of them (in their lane) have reservations, this scenario could be avoided entirely. A simple way to enforce this would be to insist that no vehicle can get a reservation unless the vehicle in front of it already has one. Unfortunately, there is no way to strictly enforce this: vehicles do not communicate their positions (and even if they did, they could be untruthful).

However, because the vehicles communicate the time at which they plan to arrive at the intersection, as well as what their velocity will be when they get there (quantities which the vehicles have no incentive to misrepresent), it is possible to approximate a vehicle's distance from the intersection, given a reservation request by that vehicle. We approximate this distance, which we call the *reservation distance*, as $v_a(t_a - t)$, where v_a is the proposed arrival velocity of the vehicle (at the intersection), t_a is the proposed arrival time of the vehicle, and t is the current time. This approximation assumes the vehicle is maintaining a constant velocity.

The policy uses the approximation as follows. For each lane i, the policy has a variable d_i, initialized to ∞. For each reservation request r in lane i, the policy computes the reservation distance, $d(r)$. If $d(r) > d_i$, r is rejected. If, on the other hand, $d(r) \leq d_i$, r is processed as normal. If r is rejected after being processed as normal, $d_i \leftarrow \min(d_i, d(r))$. Otherwise, $d_i \leftarrow \infty$.

While this heuristic does not guarantee that vehicles only get reservations if all vehicles in front of them already have reservations, it makes it much more likely. Two properties make the approximation particularly well-suited to this problem. First, if a vehicle is stopped at the intersection, its reservation distance will be

approximated as zero. No vehicle behind it will be granted a reservation before it is—no smaller reservation distance is possible. Furthermore, because the reservation distance is the product of the arrival velocity and the time until the vehicle arrives, as vehicles approach the intersection and slow down, the reservation distance gets smaller and more accurate. Thus, vehicles most susceptible to the problem described in Figure 4.5 are the most likely to be protected against it. The second property is that because the estimate uses the arrival velocity of the vehicle, it overestimates the distance of vehicles expecting to accelerate significantly before reaching the intersection. It is this expectation that causes driver agents to reserve space-time that is needed by vehicles in front of them. Note also that this heuristic only works within a single lane—each lane keeps track of its own reservation distance.

In the example of Figure 4.5, the white vehicle's rejected reservation request would shorten the maximum allowed reservation distance for its lane. This, in turn, would cause future requests by the shaded vehicles to be immediately rejected, giving the white vehicle exclusive access (within the lane) to the reservation mechanism. Once the white vehicle secured a reservation, the reservation distance would be reset to the maximum, and all vehicles would once again have equal priority.

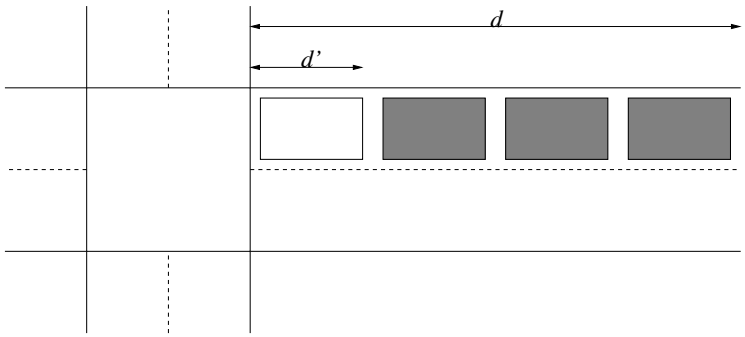


Figure 4.5: Several vehicles are waiting at the intersection. With a reservation distance of d, the front (white) vehicle is incapable of obtaining a reservation because the vehicles behind it (shaded) hold conflicting reservations. Once the white vehicle's request is rejected, the reservation distance is decreased to d'. Once the shaded vehicles cancel their reservations, the white vehicle can obtain a reservation uncontested.

Chapter 5

Driver Agent Implementation

One advantage of our intersection control mechanism is its ability to function with many heterogeneous agents. Intersection managers and driver agents can be implemented in any number of ways, provided they adhere to the protocol. In order to test the system, an implementation of each type of agent is required. The previous chapter gave some examples of intersection manager implementations. In this chapter, I describe our implementation of the other main agent in the system, the driver agent, which controls all aspects of a vehicle's motion and communication.

Our driver agent is an amalgamation of three separate agents, a *pilot*, which controls all the low-level physical motion of the vehicle, a *coordinator*, which handles all of the vehicle's communication and interaction with other agents, and a *navigator*, which selects the route the vehicle will take to reach its destination. Figure 5.1 shows the ways in which each component interacts with the environment. The three sub-agents communicate by modifying variables pertaining to approved reservation parameters in the enclosing driver agent object. The enclosing driver agent also contains a finite state machine that controls and changes state in response to the behavior of the sub-agents.

By separating the different components of the driver agent, it is possible to use only a subset of them. Because the three components are kept as disjoint as

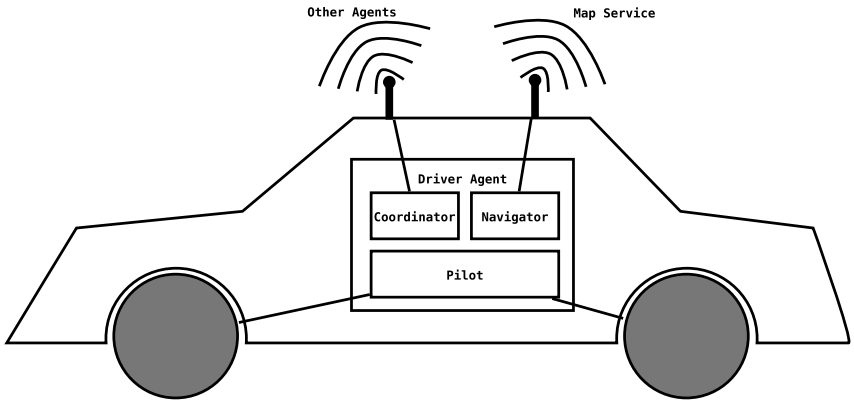


Figure 5.1: A diagram of the driver agent architecture. The pilot handles the physical motion of the vehicle, the coordinator communicates with other agents, and the navigator uses a map service to help determine routing.

possible, they do not rely heavily on each others' implementations. For example, the coordinator needs only to know the vehicle's current position and velocity, as well as the location and structure of the next intersection, to make a reservation for the vehicle. Once the reservation is made, it is up to the pilot to keep the reservation. If the coordinator determines that keeping the reservation is no longer possible, it can cancel and make a new reservation. The pilot in this case could even be a human driver, assisted by cues to speed up or slow down in order to make the reservation. Of course, the coordinator would need to understand that the pilot had limited capabilities, and make a reservation with enough leeway to be safe.

5.1 Pilot

The pilot is responsible for directly controlling the physical motion of the vehicle. Lane keeping, collision avoidance, and velocity control are all within the purview of the pilot.

5.1.1 Lane Keeping

Lane keeping is a behavior that consists solely of modifying the steering angle of the vehicle, and is thus entirely independent of the rest of the pilot's behavior. Lane keeping is active at all times—the vehicle is always attempting to stay in its current lane. The lane-keeping behavior is designed to be robust to sudden lane reassignment, and this is how both turning and lane changing are implemented: the driver agent simply changes which lane is its "current" lane, and the pilot uses the lane-following behavior to steer the vehicle into the correct lane. This process is entirely smooth, provided the vehicle is traveling at a reasonable velocity—a condition enforced by other parts of the pilot. As explained in Chapter 6.2.1, lanes are represented by a directed curve with an associated width. The curve runs down the exact center of the lane, and by keeping the vehicle evenly straddling the curve, the pilot ensures the vehicle stays centered in the lane.

The driver agent accomplishes this goal by keeping the front wheels turned toward a moving point on the segment. This point, which we call the *aim point* is farther along the segment than the vehicle. The aim point is computed by first projecting the point at the front and center of the vehicle onto the line segment, and then displacing this point in the direction of the line segment by an amount we call the *lead distance*. The lead distance is an affine function of the vehicle's velocity that is equal to a *minimum lead distance* when the vehicle is not moving. The proportional lead distance is necessary because otherwise at high velocities, the required steering angle may change faster than the pilot can steer, resulting in wildly erratic steering or the vehicle driving in circles. The minimum lead distance is to ensure that the lead distance does not get too small, the effect of which is equivalent to the velocity being too large. By ensuring the aim point is at least some distance farther down the lane, we know the vehicle will end up in a stable configuration traveling in the proper direction. Figure 5.2 depicts how the pilot determines the

lead distance (and subsequent aim point) for different velocities.

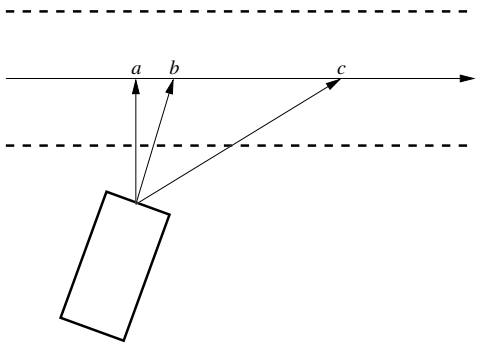


Figure 5.2: A vehicle is attempting to follow the lane. To do so, it first calculates the point that represents the projection of its position onto the directed line segment running down the center of the lane (a). Then, depending on its velocity, it displaces the resulting point in the direction of travel by a small or large amount to obtain the point at which it should aim its front wheels. For low velocities, the point will not be displaced much—only enough to ensure the vehicle moves in the correct direction (b). For higher velocities, the aim point must be farther along the lane, so that the vehicle's steering will be more gradual and thus more stable (c).

This method of lane following is of course only one possible method, and was selected because it is sufficient for our purposes. It also assumes a complete separation between determining where the lane is—a given in our simulation—and following the lane. In a real autonomous vehicle, it may be necessary to blur the line between these two processes. The reservation system's functionality does not depend on the driver agent using this particular algorithm, provided the driver agent turns within some mutually understood constraints.

5.1.2 Collision Avoidance

To avoid hitting vehicles in front of it, the pilot maintains a *following distance* from the vehicle in front of it. This following distance is a fixed *minimum following distance* of 0.5 meters, plus the amount of distance it would take for the vehicle to come to a complete stop. In the vast majority of cases, this following distance is overly conservative. However, improving it would require specific knowledge about

the velocity and deceleration capabilities of the other vehicle. In a real-world implementation, we might be able to know the velocity of the other vehicle and make some assumptions about the other vehicle's performance capabilities, but for this simulator we do not. Maintaining the following distance is the most important velocity-modifying task of the pilot, and as such, it takes precedence over all other tasks. Even if decelerating to maintain the following distance will make the vehicle miss its reservation, the pilot will still decelerate to avoid getting too near the vehicle in front of it.

5.1.3 Arriving On Time and Velocity

Secondary to maintaining a safe following distance, the pilot is responsible for ensuring that if the vehicle has a reservation, it arrives at the intersection in accordance with that reservation. If the vehicle does not have a reservation, the pilot must prevent the vehicle from entering the intersection at all. In the case of a reservation, the vehicle must arrive during a particular window of time, and there may also be restrictions on the vehicle's velocity at the time of arrival. These restrictions often take the form of a very specific velocity, but in other cases they may only define a lower bound on the vehicle's arrival velocity, to ensure that the vehicle departs the intersection by a particular deadline. As each case has different requirements on the vehicle, the pilot has three separate behaviors for meeting arrival parameters: one for arrival time only, one for arrival time and exact velocity, and one for arrival time and a minimum velocity. Each behavior must also ensure that the vehicle does not arrive at too high of a velocity in the case that the vehicle needs to make a turn at the intersection. Note that none of the behaviors decides when the vehicle should accelerate, only when it should decelerate. The default action for the pilot is to accelerate when possible.

Time Only

If the vehicle needs only to arrive at a specific time, the pilot estimates the distance the vehicle will travel if it begins decelerating at the next time step and continues to decelerate until the time of the reservation. If this distance is greater than the distance to the intersection, that means that the pilot cannot wait until the next time step to begin deceleration, or it will enter the intersection early. In this case, the vehicle decelerates. If the distance the vehicle travels is less than or equal to the distance to the intersection, then it is possible for the pilot to delay the vehicle's arrival sufficiently to avoid arriving early. In this case, the vehicle accelerates, if possible. Figure 5.3 illustrates three possible situations.

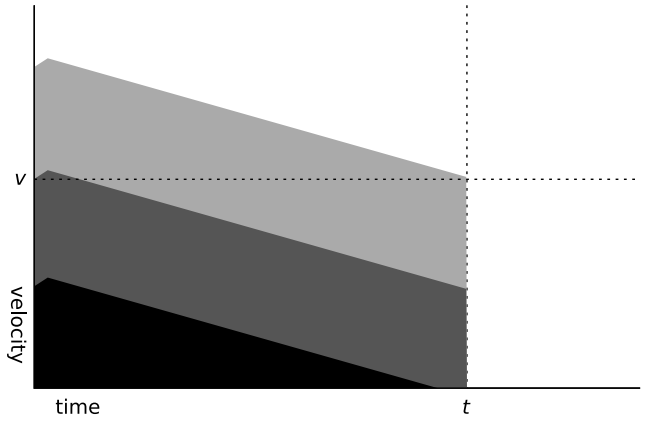


Figure 5.3: Three possible scenarios for determining whether or not to decelerate. Assume that the reservation is at time t and that the distance to the intersection is vt. If the vehicle decelerates at the next time step and follows the lightest curve, it will travel the distance under that curve, which is clearly greater than vt, and thus will arrive too early. The pilot should thus decelerate immediately. The medium curve shows the vehicle can arrive later than the arrival time, so the pilot should not decelerate. The darkest curve allows the vehicle to stop completely before the intersection. In this case, the pilot should also continue to accelerate.

Time and Exact Velocity

Ensuring that the vehicle arrives on time and with a specific velocity is somewhat more complicated than time alone. In this case, the pilot calculates the time it will take to, at the next time step, change as quickly as possible to the target arrival velocity and then maintain that velocity until arriving at the intersection. Note that "change as quickly as possible" can mean accelerating or decelerating, depending on whether the vehicle's current velocity is above or below the target arrival velocity. If the time to carry out these actions is less than the time until the reservation, then the pilot decelerates the vehicle. Additionally, if there is not sufficient time to reach a velocity at or below the target arrival velocity, the pilot decelerates the vehicle. Otherwise, the pilot accelerates.

Time and Minimum Velocity

Ensuring that the vehicle arrives on time, but at a velocity at least as great as a specified minimum arrival velocity is very similar to ensuring the vehicle arrives on time and with an exact velocity, the difference being that the pilot does not decelerate the vehicle if it believes it will arrive above the minimum arrival velocity. Recall that for ensuring arrival parameters, the pilot is not responsible for parameters that are not possible. That task is the purview of the coordinator.

5.2 Coordinator

The coordinator is responsible for all aspects of coordination with other agents, including intersection managers and other driver agents. Externally, the coordinator sends and receives messages, but internally, the coordinator also sets the arrival parameters which the pilot attempts to keep, and ensures that keeping the current arrival parameters is possible.

5.2.1 Intersection Type

The first task of the coordinator, as the vehicle approaches an intersection, is to determine what type of intersection it is. If it is a managed intersection, the coordinator knows it will need to use the V2I protocol. If it is an unmanaged intersection, the coordinator must use the V2V protocol. The type of intersection is assumed to be a part of the global map available to the driver agent, and in turn, the coordinator.

5.2.2 Determining Reservation Parameters

The most important and most difficult task of the coordinator is to select the parameters for the driver agent's next REQUEST or CLAIM, the messages that stake out space-time in the intersection in the V2I and V2V scenarios, respectively. This complex procedure involves both precise calculation as well as heuristic estimation. No agent can be entirely sure about the future, so the coordinator first makes some broad educated guesses about what will happen in the future, and then based on those guesses, precisely calculates the outcome.

Optimism and Pessimism

A naïve driver agent can perform poorly when, for example, it makes a reservation while stuck behind a slower-moving vehicle. If the vehicle in front subsequently accelerates, the driver agent should account for that by accelerating as well (possibly switching to an earlier reservation).

To account for situations like this one, we added the notion of *optimism* and *pessimism* to the coordinator. An optimistic coordinator assumes it will arrive at the intersection in the minimum possible time. A pessimistic coordinator, on the other hand, assumes it will be stuck at its current velocity until it reaches the intersection. All coordinators begin optimistic. If a coordinator has to cancel its reservation or alter its CLAIM because there is no way for it to arrive on time,

it becomes pessimistic. A coordinator which finds itself no longer stuck behind a slower vehicle will become optimistic. Due to the relatively infrequent and smooth transitions through these "moods," our coordinator agent can take advantage of improving circumstances without causing it to send excessive numbers of messages every time conditions change.

Arrival Lane

The first parameter the coordinator fixes when determining arrival parameters is the arrival lane. In all of our simulations, the arrival lane is the current lane of the coordinator's vehicle. However, if we enabled a lane-changing behavior for the driver agent, it could choose a different lane. Additionally, since a REQUEST message can contain more than one set of arrival parameters, a vehicle could generate arrival parameters for more than one lane. If the vehicle acquires a reservation in another lane that is preferable—perhaps by being earlier—it could then attempt to fulfill that reservation by first changing lanes. In a V2V scenario, only one CLAIM can be generated, and the vast majority of these situations will have only one available arrival lane, so the coordinator always chooses the current lane.

Departure Lane

In addition to choosing the arrival lane, the coordinator must choose the departure lane for the vehicle. As part of the information about the intersection, all driver agents have access to a precomputed map from arrival lanes to an ordered list of departure lanes. The departure lanes are ordered by distance from the arrival lane, where the distance from an arrival lane to a departure lane is defined as the distance from the point the arrival lane enters the intersection to the point the departure lane exits the intersection. This ordering allows vehicles to determine the shortest route through the intersection for a given arrival lane and set of candidate departure lanes. If a vehicle were turning left, the nearest lane would be the leftmost lane of

the departure road, whereas if the vehicle were turning right, the nearest lane would be the rightmost lane of the departure road. If the vehicle were not turning, the nearest lane would be the lane in which the vehicle already is traveling, followed by the lanes to the right and left of that lane, and then further outward, in a sort of one-dimensional spiral. Our coordinator has the ability to take advantage of this information if it is configured to make REQUESTs with more than one set of traversal parameters. In this case, for each set of traversal parameters beyond the first, it adds the next furthest departure lane. However, in all of our experimental data, we only allowed one set of traversal parameters per request, and as such, none of our experimental data includes this feature.

Maximum Velocity

Once the arrival and departure lanes are fixed, the coordinator can determine the maximum velocity at which it can safely negotiate the transition between the two lanes. If a vehicle takes a turn too fast, it may not be able to stay on the road, especially if the road is wet or icy. Less dramatically, it might also exit the intersection in excess of the destination lane's speed limit. The vehicle should only traverse the intersection at a velocity low enough such that it:

- can maintain control
- obeys all speed limits
- does not cause significant discomfort to its passengers

In the real world, this determination would involve a complicated mathematical computation involving road conditions, coefficients of static and kinetic friction, and other parameters beyond the scope of the simulator. In our simulator, the coordinator determines the maximum speed through a series of internal simulations that decrease the range of acceptable turn velocities. To begin with, the maximum

acceptable velocity is the minimum of the speed limits of the arrival and departure lane and the maximum velocity of the vehicle. Using a binary search–style process, the coordinator conducts an internal simulation of its vehicle making the intersection traversal at a velocity equal to the middle of the currently acceptable range of velocities. At the end of the simulation, when the vehicle exits the intersection, several quantities are measured, including the angle of departure, the distance from the vehicle to the center of the departure lane, and the steering angle of the vehicle. Based on how far these values are from their ideals—the vehicle exits exactly in the middle of the lane, facing the same direction as the lane, with zero steering angle—the trial is judged either "safe" or "unsafe". If the trial is unsafe, the maximum allowed traversal velocity is lowered to the velocity of that trial. Otherwise, the lower end of the range (which starts at zero) is increased to the velocity of that trial. The trials continue until the range is sufficiently small, at which point the lower end of the range is selected as the maximum traversal velocity for that trajectory. If zero is selected as the maximum traversal velocity, that trajectory is not allowed for the vehicle (as it would be unsafe). As an optimization in the simulator, these values are cached globally for all vehicles with identical properties.

Arrival Time

To estimate the vehicle's arrival time, the coordinator relies heavily on the optimism/pessimism heuristic. If optimistic, the coordinator assumes that the vehicle will begin to accelerate immediately to either the vehicle's top speed or the speed limit of the current lane, whichever is smaller. The coordinator also assumes that the vehicle will be able to maintain this velocity for as long as possible (or reach as near to it as possible) before needing to slow down to ensure that the vehicle does not arrive at the intersection traveling too quickly to traverse it safely. If pessimistic, the coordinator assumes that it will not be able to accelerate beyond its current velocity, but that it may still need to decelerate so as not to arrive at the

intersection at too high a velocity.

Arrival Velocity

If it is possible for the vehicle to arrive at or above the maximum traversal velocity, the coordinator will select that as the arrival velocity. Otherwise, the computation is slightly more complicated, as it has many separate cases. If the vehicle is currently traveling at a velocity below the maximum traversal velocity, and the coordinator is pessimistic, it selects its current velocity as the arrival velocity. If the coordinator is optimistic, it determines the maximum velocity the vehicle could attain if it accelerated all the way to the intersection and uses that as the arrival velocity, unless that velocity is above the maximum traversal velocity, in which case the maximum traversal velocity is used.

In previous versions of the simulator [Dresner and Stone, 2005], the arrival time and velocity prediction were not as well separated from the algorithms that attempted to maintain them. By separating prediction from control—into the coordinator and pilot, respectively—each part becomes much simpler. While this does limit the sophistication of the algorithms slightly, it makes reasoning about the correctness or performance of the entire agent much simpler.

5.2.3 Determining Possibility of Current Arrival Parameters

In addition to choosing arrival parameters, it is the coordinator's responsibility for determining if these parameters can be met given the vehicle's current state. While a vehicle can decelerate whenever it wants, acceleration may be limited by any vehicles in front of that vehicle. Due to this property, the coordinator need only ensure that the vehicle will not be too late or too slow—the pilot can ensure that the vehicle will not be early or too fast by decelerating preemptively. When evaluating whether its vehicle can arrive on time or at sufficient velocity, the coordinator assumes the best of all possible worlds. If, by accelerating immediately to its top speed or the

speed limit (whichever is lesser), the vehicle can arrive early enough and fast enough at the intersection, the coordinator deems the current arrival parameters possible. Otherwise, it either sends a CANCEL message (in the V2I scenario) or revises its CLAIM (in the V2V scenario).

5.2.4 V2V Behavior

In the V2V scenario, the coordinator still uses the same methods to calculate initial arrival parameters, but it also adds some extra behavior around this calculation. First, the coordinator does not immediately begin to transmit its calculated CLAIM. Instead, it first spends some time listening to the other vehicles' transmissions, a behavior we call *lurking*. Outside of a specified distance, called the *lurk distance*, the coordinator is entirely passive, gathering information about other vehicles' intentions. Once inside the lurk distance, the coordinator will begin transmitting its latest CLAIM repeatedly. To choose a CLAIM, the coordinator first determines its arrival parameters independently of other vehicles. Next, it determines which of the other vehicles' CLAIMs are permissible (will be allowed). The coordinator then selects the next available time at or after its calculated arrival parameters such that the resulting CLAIM will not be dominated by any permissible CLAIM about which it knows. In some scenarios, cycles in the dominance graph of the existing CLAIMs will cause this CLAIM to be nonpermissible, even though it was not dominated by any of the previously permissible CLAIMs. In this case, the coordinator finds a more restrictive CLAIM: one that is not dominated by *any* existing CLAIM, and is therefore guaranteed to be permissible. Once the vehicle has passed the point of no return (it cannot stop before entering the intersection), the vehicle commits to the CLAIM, and continues to broadcast it until it has completely traversed the intersection. A video of this process in action in the `aim3` simulator can be seen on the videos page of the project website at `http://www.cs.utexas.edu/~kdresner/aim/`.

5.3 Navigator

Of the three components in our prototype driver agent, the navigator is the only one that has already been fully realized in a modern vehicle. As such, it is not surprising that the navigator is the simplest of the three components. Global Positioning System (GPS) navigation technology exists that, given a starting location and a destination, can select a route between the two accounting for current traffic conditions, travel time, and distance. Our navigator is responsible for determining which way the vehicle will turn at an upcoming intersection. It makes this decision by calculating the shortest (time) route, calculating the time to traverse a segment between intersections based on the distance, speed limit, and vehicle's maximum velocity. It calculates intersection traversal time in the same manner as the coordinator. To optimize the search process, it uses the A* search algorithm, with a heuristic distance estimate

$$g(p) = \frac{d(p,q)}{\min(v_{max}, l_{max})},$$

where d is the Euclidian distance function, p is a point, q is the destination of the vehicle, v_{max} is the maximum velocity of the vehicle, and l_{max} is the maximum speed limit of any road [Hart *et al.*, 1968]. This method is more than sufficient for the purposes of the simulator and the small simulated world that the simulator models. In a real-world implementation, such computation could occur remotely (perhaps as a service), or involve more sophisticated algorithms. Because of the infrequence with which routes are calculated and the high tolerance for latency—a 10-second delay for an optimal route is perfectly acceptable—the specifics of the implementation are not as critical as the other components. Nonetheless, there is a large space of potential solutions, including multiagent methods, each with its own benefits and drawbacks. Further discussion of such methods is beyond the scope of this thesis.

5.4 Human Driver Agent

In some experiments, we use what we call a "human" driver agent, which is meant to simulate the important properties of a human driver sharing the road with autonomous vehicles. This driver agent is very similar to the autonomous driver agent described above, except that it is incapable of wireless communication or any prediction short of that required for distance-keeping. Whereas the autonomous driver agent understands the periodicity of traffic signals and can predict the next "green" cycle, the human driver acts only based on the current state of signals. If the signal is green, it proceeds, if it is yellow or red, it stops if it can, otherwise it proceeds. The human driver agent also keeps a longer following distance than the autonomous vehicles. In all other aspects, namely those regarding steering and lane keeping, it is identical to the autonomous driver agent.

Chapter 6

Simulator

There are many traffic simulators available for research purposes. Some of these simulators are designed to model vehicle kinematics with extremely high fidelity, including tire friction, engine power output, and even aerodynamics. Others deal with very large networks of roads or freeways, or model traffic flow instead of individual vehicles [Sukthankar *et al.*, 1995; Helbing *et al.*, 2001]. Many simulators are designed to model true human behavior, rather than testing custom agent algorithms [Caliper Corporation, 2009]. When this research began, however, none gave us the ability to easily replace the mechanism by which intersections are governed. Since that is the main focus of this work, we required a custom simulator. Furthermore, we needed a simulator that could simulate individual vehicles, but we did not need extremely high fidelity. Rather, we need to be able to simulate a very large number of individual vehicles, and examine the effect of the vehicles' interactions at intersections on the traffic system as a whole, and thus required a simulator that sits somewhere between the two extremes. Because we were unable to find an "off-the-shelf" simulator that gave us both the flexibility and the precision that we needed, we built a custom time-based simulator for our experiments. This simulator has evolved into a major contribution of the research presented in this thesis, and the source code will be released upon Publication.

6.1 A Brief History Of The AIM Simulator

Throughout the research presented in this thesis, our custom simulator has gone through three major revisions. In the remainder of this chapter, I describe the most recent and full-featured simulator in detail. However, here I discuss and present the earlier versions, as some of our experimental results were obtained using the earlier versions, each of which has various limitations. Furthermore, many of the previous publications of this research used previous incarnations of the simulator.

6.1.1 The First Simulator

The first version of our simulator, `aim1`, models a single four-way intersection of perpendicular roads. Vehicles cannot turn at the intersection—the driver agent controls only the vehicle's velocity. The main purpose of this simulator was to conduct a proof-of-concept experiment to determine roughly how well the reservation-based intersection control mechanism compared to a theoretical traffic signal or stop sign. Communication between agents was synchronous and handled by a method call, with the driver agents calling methods directly in the intersection manager, of which there was only one. In addition to being unable to turn, vehicles in the first simulator were required to maintain a constant velocity while in the intersection. No V2V communication or coordination was possible.

6.1.2 The Second Simulator

The second version of the simulator, `aim2`, added the ability for vehicles to turn and accelerate while in the intersection. By enabling vehicles to accelerate in the intersection, the simulator could use other intersection control policies besides FCFS, including emulation of a traffic signal and stop sign—both of which require vehicles to be able to stop at the intersection and then accelerate. The second version also added the capability of an intersection manager to control physical signals with a

signal model, allowing for policies such as FCFS-Signal (see Chapter 8). With FCFS-Signal, the simulator gained the ability to mix (simulated) human-driven and autonomous vehicles.

6.1.3 The Current Simulator

The current incarnation of the simulator, dubbed `aim3`, supports all of the features of the first two versions, as well as the capability of simulating and managing multiple intersections, parameterized vehicles, vehicle-to-vehicle communication, variable reliability communication, arbitrarily large queues of policies, piecewise-linear lanes, and non-rectangular intersection geometries. In the rest of this chapter, I discuss in detail many of the key elements of this simulator: the modeling of the physical and geographic layout of the roadways, the vehicles themselves, the communication model, physical motion, statistics, the overall main loop, and visualization.

6.2 Layout

The part of the simulator that models the infrastructure is the *layout*. The layout is like a map, containing all the roads and lanes, along with all the important information about the geometry of those lanes and roads. Such a layout could be constructed from a predefined format such as DARPA's Route Network Definition File (RNDF) [DARPA, 2007b], however it is important to note that this section describes an internal representation, optimized for use by the simulator—it would not be practical to store the layout internally as RNDF, because anytime an element of the simulator wanted to access information about the layout, the RNDF content would need to be parsed and analyzed. However, using a format like RNDF as an input would allow us to define intersections more arbitrarily, but with less automation. While the simulator could be extended to accept RNDF as an input or to define certain physical structures more freely, it does not currently have that capability..

6.2.1 Lanes

A *lane* is just that – a model of a lane of traffic. In the simulator, every lane has three properties, a starting point, an ending point, and a width. Additionally, every lane may have another lane that leads into it, out of it, or borders it to the right or left. This "chaining" allows lanes to be created from multiple substituent lanes. Using only a few base lane implementations, any number of complex lanes can be created. The only current lane implementation is a line segment lane, so any lanes must be piecewise linear, however, the interface is such that a lane could be made from a wide variety of functions such as a cubic curve. Connecting lanes on the left and right sides allows the creation of multiple-lane roads on which vehicles can change between the adjacent lanes. In addition to physical properties, lanes also have a speed limit and a traffic level. The traffic level, measured in vehicles per second, determines if and how vehicles are spawned in that lane.

6.2.2 Roads

A *road* is a group of lanes that travel together, in the same direction. At the heart of the road is a list of lanes, ordered from leftmost to rightmost. Each lane is adjacent to the lanes before and after it in the list. Because all lanes in a road must travel in the same direction, if more than one direction of travel is needed, more than one road is required. For this reason, most roads also have a *dual*—a separate road that runs parallel, but in the opposite direction. If road R is the dual of road R', then R' is the dual of R. Thus, most two-way streets are implemented as two roads that are duals of one another. While we assume throughout all of our experiments that vehicles drive on the right, this implementation of roads ensures that this need not be the case. As long as the two roads are duals of one another, everything in the algorithm will work correctly.

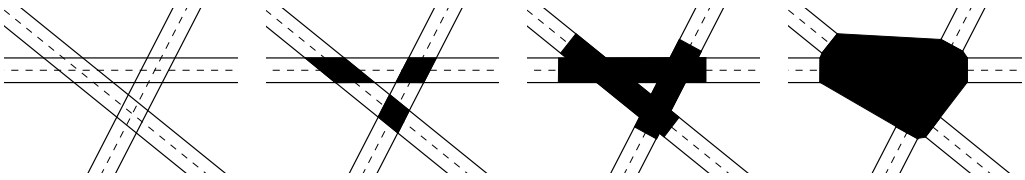
6.2.3 Defining Intersections

Once roads are defined, we can define intersections of roads. The simulator allows for the intersection of any set of roads—the set of roads is the only thing from which the intersection is created. The creation of an intersection from a set of roads proceeds in three steps, which are illustrated in Figure 6.1.

1. Finding the geometric intersection of all the lanes in all the roads, or more specifically, the union of the pairwise geometric intersections of all lanes.

2. Extending the intersection by including the area of every lane from a fixed distance before any part of that lane enters the intersection until a fixed distance after any part of that lane leaves the intersection. This extension ensures that lanes always enter the intersection perpendicular to its boundaries, prevents vehicles from getting too close to one another before entering the intersection, allows edge tiles to control only entering and exiting vehicles, and handles cases in which a strict intersection would leave the intersection in several disjoint pieces that lanes may enter and exit multiple times.

3. Extending the intersection to be its own convex hull. This process eliminates any remaining "holes" that might be in the intersection, as well as smoothing out some of the angles that might make for impossibly sharp turns.



(a) Initial set of roads (b) Strict intersection (c) Extension (d) Convex hull

Figure 6.1: The intersection construction process. The initial set of roads is shown in 6.1(a). Steps 1, 2, and 3, are shown in 6.1(b), 6.1(c), and 6.1(d), respectively.

6.3 Vehicles

At the very simplest, a vehicle is a rectangle with a fixed length and width—the simulator does not attempt to model the third dimension (height). But vehicles in the simulator have a lot of other fixed properties, as well as variable states, in order to make them more useful for experimentation. The fixed properties include the location of the front and rear axles and the performance capabilities of the vehicle. State variables include velocity, acceleration, and absolute position. On vehicles equipped with a computerized driver agent, more properties may be present, such as a communication range and queues of messages waiting to be sent or just received. Despite being just a rectangle, the vehicle model and its associated code are the single most complicated component of the simulator. Here, I describe this component in great detail.

6.3.1 Vehicle Properties

At a bare minimum, vehicles in the simulator have the following fixed properties:

Vehicle Identification Number (VIN): a unique numeric identifier for the vehicle

Length: the distance from the front of the vehicle to the rear of the vehicle, in meters

Width: the distance from one side of the vehicle to the other, in meters

Front Axle Displacement: the distance from front of the vehicle to the front axle, in meters

Rear Axle Displacement: the distance from front of the vehicle to the rear axle, in meters

Maximum Velocity: the maximum speed at which the vehicle can travel, in meters per second

Maximum Acceleration: the maximum rate at which the vehicle can accelerate, in meters per second squared

Minimum Acceleration: the maximum rate at which the vehicle can decelerate, in meters per second squared (this is a negative quantity)

Maximum Steering Angle: the maximum angle away from center that the vehicle can turn its front wheels in either direction, in radians

Maximum Steering Rate: the maximum angular velocity at which the vehicle's steering angle can be altered, in radians per second

Sensor Range: the furthest distance from the vehicle that its sensors can detect another vehicle, in meters

Transmission Range: the distance that the vehicle can transmit a wireless signal, in meters

The constants representing the distance from the front of the vehicle to the front and rear axles allow more accurate simulation of vehicle turning. Specifically, they allow the simulator to treat different styles of vehicle differently. The distance between the front and rear axles is known as the *wheelbase*. Vehicles with shorter wheelbases can turn more sharply than those with longer wheelbases—if the simulator is to accurately model turning, it needs access to these important parameters. Furthermore, a vehicle with a long hood will turn differently than a vehicle whose front wheels are located nearer to the front of the vehicle.

The maximum steering rate limits the ability of a driver agent to turn the wheels over time. Even a computerized driver will be limited by the hardware in the vehicle, and will not be able to adjust the vehicle's steering infinitely fast.

This limitation more accurately approximates vehicle turning, including some of the more dangerous aspects. If the driver cannot turn the wheels instantaneously, it must ensure that it does not drive around corners at too high a velocity—it may not be able to straighten out quickly enough and wind up veering off the road instead.

We did not experiment with variable transmission ranges, so for all experiments, our transmission range was 250 meters.

All vehicles also have the following state variables:

Position a Cartesian pair of (x, y) coordinates

Velocity the current forward velocity of the vehicle, in meters per second

Heading the absolute angle (measured from the positive X-axis) of a ray from the center rear of the vehicle to the center front of the vehicle, in radians

Acceleration the current rate of forward acceleration of the vehicle, in meters per second squared

Steering Angle the angle to the left of center to which the front wheels of the vehicle are turned, in radians

The simulator uses a Cartesian coordinate system that could be adapted to use latitude and longitude via a simple transformation. In this coordinate system, east is the direction of the positive X axis, and north is the direction of the positive Y axis. A vehicle with a heading of zero radians would be driving due east, and a vehicle with a positive steering angle $0 < \psi < \frac{\pi}{2}$ would be turning to the left.

6.3.2 Driver Agent Access To Vehicle Properties And State

Providing driver agents with access to vehicle constants is trivial—they can access this information at any time and have accurate and precise knowledge of these quantities. Providing access to the variable state is slightly more complicated. To

handle this interplay between driver and vehicle, we provide each vehicle with a set of *gauges*. A gauge is a window by which the driver can view the state of the vehicle, but not always with high accuracy or precision. Depending on how the gauge is instantiated, it can include several types of noise, as well as be entirely inoperable. In addition to providing the driver agent with access to internal state, gauges also allow driver agents to read information from simulated external sensors. These sensors include a basic interval sensor that reports the distance to the next vehicle in the current lane, and a more complicated simulated laser range finder that reports the distance and angle to the closest object in the vehicle's view. Because simulating the laser range finder is computationally expensive, and because the interval sensor is sufficient in most situations, the vehicle can turn off the laser range finder when it is not needed. If it is off, the laser range finder is not simulated.

There are several facilities in the simulator for driver agents to modify the state of the vehicle. While clearly a driver agent should not be able to set the position of a vehicle, it can alter the steering and acceleration, as in a traditional vehicle. To steer, the driver sets a desired steering angle, and the vehicle moves the wheels over time toward that steering angle. To change velocity, a driver agent has several options. It can specify a target velocity—from which the vehicle will infer the required acceleration in order to reach that target velocity as quickly as possible—or it can specify both a target velocity and an acceleration, in which case the vehicle will accelerate at the given rate until it reaches the target velocity.

6.3.3 Vehicle Sensor Data

In addition to gauges that give the driver agent views of the internal state of the vehicle, there are gauges that give the driver agent information about what is going on outside the vehicle. These gauges are designed to provide information from simulated sensors that the vehicle could have. While an actual autonomous vehicle would have a multitude of outward-facing sensors, including laser range finders,

short-wave radar, lidar, and video cameras, many of these technologies are either very difficult to simulate or do not make sense in our simulated environment. To operate safely in our simulated environment, we have determined that a vehicle really only needs to sense one thing: how far away the next vehicle in front of it is. It may not be well-defined as to which vehicle is the next vehicle in front, and so we created two different sensors that try to accomplish this: a simplified simulated laser range finder that can be used in any situation, and an interval sensor that is much cheaper to use computationally, but can only be used when the vehicle is traveling within a lane.

Simulated Laser Range Finder

Modern laser range finders and distance sensors can provide a large amount of distance and angle data to a mobile agent. In a real life setting, this information would definitely prove useful in fine-tuning a driver agent. However, in our simulation, we must process sensor information for *all* vehicles simultaneously, and accurately simulating a full laser-range finder is not feasible. Thus, we use a simple, yet pertinent sensor reading which the driver agent can use to control its actions with respect to the other vehicles. A purely straight-ahead sensor suffices when vehicles are traveling only in straight lines. However, when a vehicle turns, it must also take into account what is going on in the direction in which it is turning. To complicate matters, when a vehicle is turning it must still take into account what is going on directly in front of it because at any point it might straighten its wheels and continue on its current heading. A sensor that points in the same direction as the wheels will not be sufficient because vehicles coming out of turns may run into vehicles ahead of them. Instead, our sensor's scope widens in the direction of the turn, while narrowing slightly from the other side. Figure 6.2 shows a scenario that demonstrates the concept. As a testament to the sensor's usefulness, when sent through an intersection without coordination of any kind, vehicles equipped with only this sensor

are able to avoid many collisions in the intersection, even with moderate amounts of traffic. The main drawback of this approach is that, unamortized, it requires $O(n^2)$ distance calculations just to determine which vehicles are in range of the sensor.

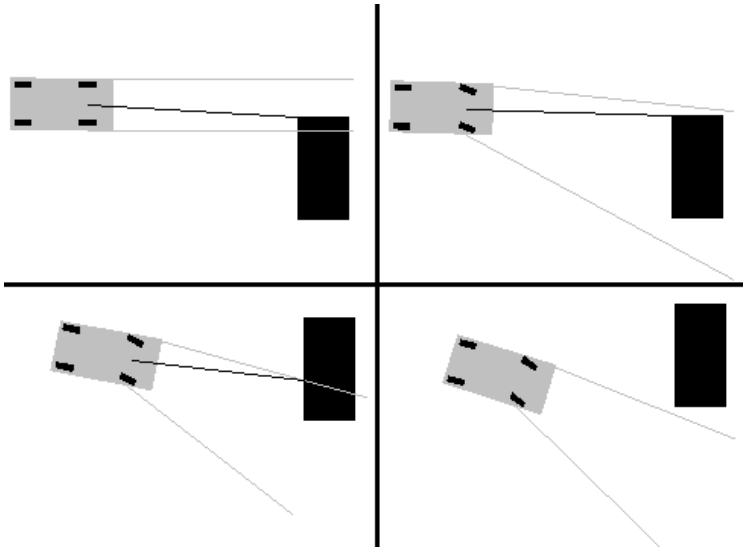


Figure 6.2: A depiction of the simulated laser range finder sensor model for the driver agents. The sensor is focused between the gray lines and does not provide information outside of them. The black line represents the reading provided to the driver agent.

Interval Sensor

Most of the time, the full simulated laser range finder will not be necessary. Vehicles spend the vast majority of their time in lanes, and when they are in the intersection—at least in the V2I scenarios—do not usually need to worry about using their external sensors. In order to make the simulation faster, we allow the driver agent to disable the simulated laser range finder, and instead use an even simpler interval sensor, which reports the distance to the next vehicle in the current lane. To provide input for this sensor, the simulator creates a list of vehicles for each lane, and then sorts those vehicles by their distance from the start of the lane. Note that it is possible for

a vehicle to be in more than one of these lanes if, for example, it is changing lanes. Vehicles inside intersections are not included. Once each of these lists of vehicles is sorted, the distances between the successive vehicles are calculated and recorded in the vehicles' interval sensor gauges. Instead of requiring $O(n^2)$ time to complete, as with the laser range finder, this process takes only $O(n \log n)$ time. Instead of only being able to simulate tens of vehicles in real time, we can simulate hundreds.

6.3.4 Vehicle Disabilities

In addition to all the things that vehicles can do, the simulator has facilities for designating what vehicles cannot do. As part of the failure mode analysis, we added the ability for vehicles to have various *disabilities*, each of which prevents the vehicle from taking certain actions. There are 8 disabilities:

NO_BRAKES The vehicle cannot decelerate.

STUCK_BRAKES The vehicle's acceleration is set to its minimum value, with a target velocity of zero and cannot be adjusted.

NO_ACCELERATOR The vehicle cannot accelerate.

STUCK_ACCELERATOR The vehicle's acceleration is set to its maximum value and cannot be adjusted.

LOCKED_STEERING The vehicle's steering angle cannot be altered.

PULLS_RIGHT The vehicle's steering angle is set to its minimum value and cannot be altered.

PULLS_LEFT The vehicle's steering angle is set to its maximum value and cannot be altered.

CRASH The vehicle's velocity and acceleration are set to zero, and the vehicle's acceleration cannot be altered.

6.3.5 Vehicle Statistics

Vehicles also store statistics for the simulator. These statistics include the amount of information sent and received, the number of intersections traversed, the total distance traveled, and the total delay experienced by that vehicle.

Communication Statistics

Associated with each message type is a method for computing the size of that message. Since some messages have variable-length fields, different messages of the same type may have different sizes. Because bandwidth is not an unlimited resource, especially in a wireless network, it is important to keep track of the amount of information traveling between vehicles to ensure that the scenario is realistic. Whenever a message is sent or received by a vehicle, the size of the message is added to the log. These quantities are then folded into global totals when the vehicle is removed from the simulation.

Delay

In Chapter 2.1, we briefly introduced *delay*—the difference in travel time for a vehicle due to the presence of the intersection and other vehicles. In the simulation, this quantity is measured on a step-by-step basis. At each step, the simulator determines what the ideal speed of the vehicle should be. The ideal speed is a function of the velocity at which it left the previous intersection, the speed limit of the current lane, and the maximum allowed turn velocity at the next intersection. A vehicle's ideal velocity is bounded above by these quantities. More specifically, if a vehicle has just departed an intersection at velocity v, its current velocity cannot exceed $v + a_{max}t$, where t is the time since it departed that intersection, and a_{max} is the vehicle's maximum acceleration. Similarly, if a vehicle cannot arrive at an intersection moving faster than v, then its velocity cannot exceed $v + a_{min}t$, where t is the time until it arrives at the intersection, and a_{min} is its minimum acceleration (maximum

deceleration). Clearly, the vehicle's speed is bounded above by the speed limit of the current lane. By taking the minimum of these quantities, we can establish the ideal velocity of the vehicle. While inside an intersection, the ideal velocity of the vehicle is calculated similarly, substituting the maximum turn velocity for the speed limit. Using this ideal velocity, we can determine the ideal amount of time it would take to cover the distance that the vehicle actually covered in that time step. The difference between this ideal amount of time, and the actual length of the time step is the delay experienced by this vehicle in this time step. Because the frequency of driver agent actions is limited by the frequency of the simulator (50 Hz), a very small amount of delay may accumulate as an artifact of the discretized time in the simulator.

6.3.6 Vehicle Archetypes

In order to simulate a variety of vehicles, the simulator includes five predefined vehicle archetypes: the coupe, sedan, sport/utility vehicle (SUV), van, and bus. The physical characteristics and performance capabilities of these vehicles are taken from representative real-world vehicles or estimated where that information was not readily available. Table 6.1 shows these properties and capabilities.

Name	l	w	v_{max}	v_{min}	a_{max}	a_{min}	d_f	d_r	ψ_{max}	$\frac{\partial \psi}{\partial t}_{max}$
Coupe	4	1.75	60	-17	4.5	-15	1	3.5	$\frac{\pi}{3}$	$\frac{\pi}{2}$
Sedan	5	1.85	55	-15	3.25	-13	1.2	4	$\frac{\pi}{3}$	$\frac{\pi}{3}$
SUV	5.131	2.007	52	-13	3.83	-13	1.18	4.126	$\frac{\pi}{3}$	$\frac{\pi}{3}$
Van	5.385	2.014	45	-10	3.08	-10	0.58	4.085	$\frac{\pi}{3}$	$\frac{\pi}{3}$
Bus	15	3	35	-9	1.3	-8	1.5	12	$\frac{\pi}{4}$	$\frac{pi}{3}$

Table 6.1: The different vehicle archetypes defined in the simulator. Properties and performance capabilities are taken from real vehicles where readily available and estimated otherwise. The properties are length (l), width (w), maximum velocity (v_{max}), minimum velocity (v_{min}), maxmium acceleration (a_{max}), minimum acceleration (a_{min}), front axle displacement (d_f), rear axle displacement (d_r), maximum steering angle (ψ_{max}), and maximum steering rate ($\frac{\partial \psi}{\partial t}_{max}$).

In addition to these built-in types, the simulator also supports any custom vehicle type, for which all of these values may be changed. Unless specially configured otherwise, when generating new vehicles, the simulator uses a provided distribution to probabilistically select from the main vehicle archetypes. For our simulations, we use only the coupe, sedan, SUV, and van, and generate them with uniform probability.

6.4 Communication

All communication in the simulator is simulated as if it were point-to-point communication. Broadcast communication, which is used in the V2V scenarios, is simply built on top of point-to-point communication: an outgoing broadcast message is sequentially delivered to all other vehicles in range as if they were the destination. There is no simulated underlying network architecture for relaying messages to an ultimate destination. While such a system is certainly worth investigating for a real-world application, in the simulator it would need to be an explicit part of the agent behavior.

Each agent that can communicate has two queues of messages, an *inbox* and an *outbox*. In the actual implementation, vehicles have a separate set of queues for V2V and V2I protocol messages, but it is equivalent to think of them as one set of queues. Whenever an agent wants to send a message, it places it in the outbox. Synchronously, the simulator examines all agents' outboxes, takes any messages in them, and then conditionally delivers them to their destinations' inboxes. The next time the destination agents are able to act, they can examine their inboxes and take actions based on the messages present. Whether or not an individual message is delivered is a function of two things: the transmission strength of the sending agent, and the distance between the sending agent and the receiving agent. The location of an intersection, for these purposes, is the centroid of the intersection's

area. For all of our experiments, we use a very simple function: the message is delivered if and only if the message strength is greater than or equal to the distance between the agents. It would be trivial to modify this function to do something more sophisticated, such as dropping the messages probabilistically as the distance grew beyond the transmission strength. Because the messages are delivered all at once in one phase of the simulation, messages sent in one time step do not arrive until the next time step.

One nice result of explicitly modeling communication (instead of using simple function calls, as in previous versions of the simulator) is that it allows us to do a *mixed simulation*. In a mixed simulation, one or more of the vehicles in the simulator is an actual physical vehicle. Each real vehicle corresponds to a proxy vehicle in the simulator whose state—position, velocity, and so forth—are continuously updated using data from the real vehicle. The real vehicle's sensors are fed information from the simulator to make it appear to the real vehicle that the simulated vehicles are real. This enables us to run experiments involving real vehicles without risking expensive damage to the real vehicles should something go awry [Nimmagadda, 2009].

6.5 Physical Motion

At each time step, the simulator must update the position of every vehicle. Because we model only planar vehicle kinematics and not dynamics, we must make a few assumptions. First, we assume that vehicles do not skid on the road. Second, we assume that vehicles move according to the following differential equations for non-holonomic motion:

$$\frac{\partial x}{\partial t} = v \cdot \cos(\phi)$$
$$\frac{\partial y}{\partial t} = v \cdot \sin(\phi)$$

$$\frac{\partial \phi}{\partial t} = v \cdot \frac{\tan \psi}{L}$$

In these equations, x, y, and ϕ describe the vehicle's position and orientation, v represents the vehicle's velocity, ψ describes the vehicle's steering angle, and L is the vehicle's wheelbase. We solve these equations holding v and ψ constant for each time step.

6.6 Global Statistics

Each vehicle maintains some statistics for itself, but the simulator as a whole also tracks several statistical quantities:

Completed vehicles the number of vehicles that have completed their journeys.

Total delay the total delay experienced by all vehicles.

Step delay the total delay experienced by all vehicles in the last time step.

Intersection traversals the total number of intersection traversals that have been made.

Data transmitted the number of bytes of data that have been sent.

Data received the number of bytes of data that have been received.

With the exception of step delay, each of these quantities is a total for all vehicles that have completed their journeys and been removed from the simulator. In order to enhance the visual display of the simulator with current views of these statistics, a *sliding window* system tracks each of these quantities as they change over time. The size of the sliding window is modifiable, but is usually set at 30 seconds of simulated time. Thus, for each of these statistics, we can always examine an average of these values over the last 30 seconds of simulation.

6.7 The Main Loop

The main loop of the simulator consists of 6 steps. Each loop of the simulator represents a discrete amount of time t, usually 0.02 seconds.

1. **Spawn Vehicles** For each lane with a traffic level $\lambda > 0$, a vehicle is generated with probability $p = t\lambda$. This vehicle is placed in a queue of vehicles waiting to spawn in that lane. Then, if there is room in the lane for a vehicle (including room for it to come to a stop), the first vehicle in the queue for that lane is spawned at the start of the lane. The traffic spawning in that lane thus roughly corresponds to a Poisson process with rate parameter λ. The major difference is that the simulator will never spawn vehicles so close together that they cannot avoid a collision. It will, however, store the generated vehicles in the queue and spawn them later, such that the overall number of vehicles spawned will be the same as if it were a true Poisson process.

2. **Provide Sensor Input** For each vehicle, that vehicle's velocity, acceleration, heading, and position are recorded to the speedometer, accelerometer, compass, and position gauges, respectively. Because the gauges themselves provide any error or noise to the readings, the true values of these quantities are used. Additionally, the interval gauge and simplified laser range finder are simulated, and the results are recorded to the appropriate gauges in the vehicle.

3. **Agent Action** Driver agents and intersection managers are given a chance to act. For driver agents, this includes reading any waiting messages, sending messages, and changing the steering angle or acceleration of the vehicle. For intersection managers, this involves reading any waiting messages, and allowing intersection control policies to act, which could in turn result in messages being sent back to vehicles.

4. **Communication** For each agent, any messages in that agent's outgoing messages queue are conditionally delivered to their destinations.

5. **Move Vehicles** The positions, velocities, and headings of all vehicles are updated based on the simulator's physical model.

6. **Collision Detection** If enabled, each pair of vehicles is examined to determine if they overlap. If so, both vehicles have their velocity set to 0 and are marked with the disability CRASH.

7. **Cleanup** Any vehicle that has traveled outside the simulated area and has arrived at its intended destination is removed from the simulation and any statistics it was keeping are merged with the global statistics in the simulator.

6.8 Visualization

Visualization is an important part of debugging the simulator, as well as making results easier to interpret. To these ends, we created a visualizer for the simulator that displays lanes, roads, intersections, and vehicles, as well as all communications in the simulator. Figure 6.3 shows a screenshot of part of the simulator's graphical display. The visualizer also shows current statistics for the simulation, both overall and using the sliding windows, but the display would be illegible in a figure, and as such they are not shown here.

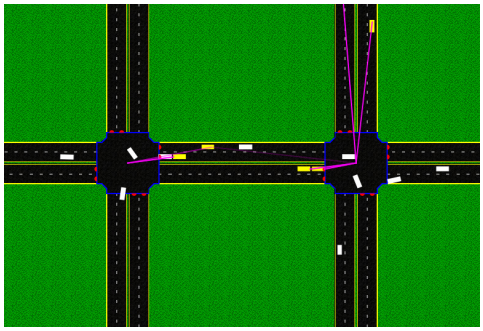


Figure 6.3: A screenshot of the simulator in action, best viewed in color. White cars are those with confirmed reservations, while yellow cars are those without. The purple lines show point-to-point communications. In scenarios with broadcast communication, an expanding purple ring shows a broadcast transmission.

Chapter 7

Experimental Results

The previous chapters introduced a novel reservation-based intersection control mechanism for autonomous vehicles. In this chapter, I present results from experiments that test all of the features introduced in the preceding chapters and demonstrate that the reservation system has the capability to improve the travel time of vehicles using it tremendously, as compared to traditional mechanisms such as stop signs and traffic signals. Our experiments evaluate the performance of the reservation system using different intersection control policies, amounts of traffic, granularities, levels of human drivers, and the presence of emergency vehicles. I first compare the system using FCFS to traffic signals of varying cycle periods using the `aim1` simulator. I then show results from `aim2`, including the stop sign control policy as implemented under our protocol, comparing these results to those from the traffic signal experiments. Next, I experiment with allowing vehicles to turn from any lane—something that would be extremely dangerous without the reservation-based mechanism. Next, I define and evaluate a simple extension to FCFS: FCFS-Emerg. Finally, I present results from a vehicle-to-vehicle (V2V) scenario, with no intersection manager. Videos of many of these scenarios running in the simulator can be viewed at the AIM project's website at `http://www.cs.utexas.edu/~kdresner/aim/`, by selecting the videos page.

7.1 The Delay Metric

In previous chapters, I have briefly mentioned the main metric used to evaluate our protocol and agent algorithms: *delay*. Delay is the increase in travel time due to the presence of the intersection manager and any congestion from other vehicles. If a vehicle could travel from its point of origin to its destination in time t without other vehicles or the intersection to deal with, but it takes time t' with those elements present, then the delay experienced by the vehicle is $t' - t$. Scientists and traffic engineers use many different metrics to gauge the performance of intersection management mechanisms like traffic signals, stop signs, and roundabouts.

One alternate metric is *throughput*. Throughput measures the maximum number of vehicles per lane that can pass through the intersection in a given period of time. While this is a useful metric, it only captures what happens at the extremes of the intersection's capabilities. An intersection may have an extremely high throughput, but it is quite possible that even with low traffic, vehicles take quite a long time to cross the intersection. By using delay, we can measure the capabilities of the intersection management mechanism at many different parts of the traffic level spectrum. In our experiments, the rate parameter for each lane, λ, is very closely related to throughput. For most experiments, the two are equal. As long as the delay is not growing without bound for a particular setting of λ, we can say that the throughput for the intersection is at least λ. Once traffic starts to back up without bound, we can say that λ has exceeded the throughput of the intersection manager. Instead, the metrics should be Delay and throughput can be thought of as analogous to *latency* and *bandwidth* in computer networking. While decreased network quality usually affects both, the two measure different things. By measuring delay across a wide variety of traffic levels, I believe we can get a better picture of an intersection's efficiency than we could with just throughput. The point at which the delay begins to increase asymptotically will indicate the

maximum throughput of the intersection.

A second metric, which we have considered explicitly in some previous publications, is *total accumulated acceleration* [VanMiddlesworth *et al.*, 2008]. This metric measures how much accelerating and decelerating the vehicle does. While not necessarily correlated with the other metrics, total accumulated acceleration can provide insight into how fuel-efficient an intersection control mechanism is. A vehicle that spends less fuel accelerating and wastes less energy by braking will use less fuel than the same vehicle would otherwise. In addition, a vehicle that does less acceleration and braking will be more comfortable for its passengers. Acceleration does not make sense as a primary metric, as the optimal solution is for no vehicle to move at all. However, as a secondary metric, total accumulated acceleration can be useful to demonstrate some of the added benefits of an intersection control mechanism.

I believe delay is the most appropriate metric for this work, especially when measured for a wide variety of traffic levels. For this reason, I will focus almost exclusively on delay in this and the following chapters. Note that it may not be useful to directly compare the metrics as measured in the simulator with real-world values, as the simulator is not designed to replicate the exact constants of the real world—values may be off by a constant factor. Instead, we use them to compare various mechanisms and policies within the simulator.

7.2 Low-Granularity-Ratio FCFS vs. the Traffic Signal

The simplest implementation of FCFS has granularity ratio 1—the entire intersection is a single reservation tile. While only one vehicle may be in the intersection at a time, if that vehicle is traveling sufficiently fast, the total amount of time for which it will occupy the intersection is small. If we increase the granularity ratio to 2, the intersection is no longer entirely exclusive. For example, non-turning vehicles

traveling north no longer compete for the same reservation tiles as non-turning vehicles traveling south (similarly, eastbound and westbound non-turning vehicles no longer compete). Here we present our initial results comparing these two instances of the reservation mechanism and several incarnations of a traffic signal.

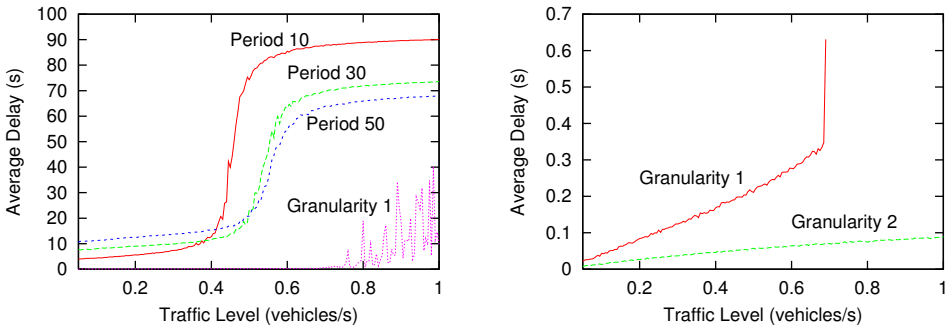
7.2.1 Experimental Setup

These experiments were carried out using the first version of the simulator, `aim1`, which is fully described in an earlier publication [Dresner and Stone, 2004]. In this version of the simulator, vehicles are not allowed to turn or accelerate while in the intersection. These restrictions do not detract from the core challenge of the problem, and the results are relevant even when the restrictions are relaxed. Each simulation contains one lane traveling in each direction, the speed limits of which are 25 meters per second. Traffic spawning probability varies from 0.0001 to 0.02 in increments of 0.0001, and each configuration runs for 500,000 steps in the simulator, which corresponds to approximately 2.5 hours of simulated time.

7.2.2 Results

Figure 7.1(a) shows delay times for traffic signal systems with varying periods, ranging from extremely short (10 seconds) to fairly long (50 seconds). As expected from real-life experience, short-period traffic signals control light traffic well, while traffic signals with longer periods work better in heavy-traffic scenarios. When traffic is sparse, a short period allows vehicles to wait a shorter time before getting a green signal. In many cities, traffic signal periods are shortened during early hours of the morning to take advantage of this fact. In scenarios with more densely packed vehicles, the per-vehicle costs of slowing to a stop and accelerating back to full speed, as well as the intervals needed to clear out the intersection (the time during which there is a yellow signal, or all signals are red), tend to dominate. This makes the longer-period signals better in these situations. In the Figure 7.1(a),

above a certain traffic level, each of the traffic signal systems reaches what appears to be a maximum delay level. This is an artifact of the simulator—when the traffic level gets high enough, the vehicles back up so far that the simulator cannot keep track of them (it cannot spawn new vehicles, for lack of a place to put them). At this point, vehicles are arriving at the intersection faster than the traffic signals can safely coordinate their passage. Thus, the point at which the delay spikes upwards indicates the maximum throughput of each traffic configuration.



(a) FCFS vs. Traffic Signal

(b) Increasing Granularity Ratio

Figure 7.1: 7.1(a) shows average delays for traffic signal systems with period 10, 30, and 50 seconds plotted against varying traffic levels along with a 1-tiled reservation-based system. 7.1(b) shows average delays for granularity-ratio-1 and 2 FCFS policies with varying traffic levels. Spawning probability was varied in increments of 0.0001, and each configuration was run for 1,000,000 steps of simulation (approximately 5.5 hours of simulated time). Each direction has 1 lane.

Also in Figure 7.1(a) are the delays for the granularity-ratio-1 and 2 FCFS policies. With the car spawning probability below about 0.013, the granularity-ratio-1 policy's delay is visually indistinguishable from the x-axis, while this is true for the granularity-ratio-2 reservation system for the whole graph. Figure 7.1(b) shows the bottom 0.7% of the graph, enlarged to show these results in more detail. At the vehicle spawning rate of 0.02, all of the traffic signal systems are already beyond maximum capacity, while the granularity-ratio-2 system is allowing vehicles through without even adding a tenth of a second to the average vehicle's travel time.

7.2.3 The Effects Of Poisson Arrivals

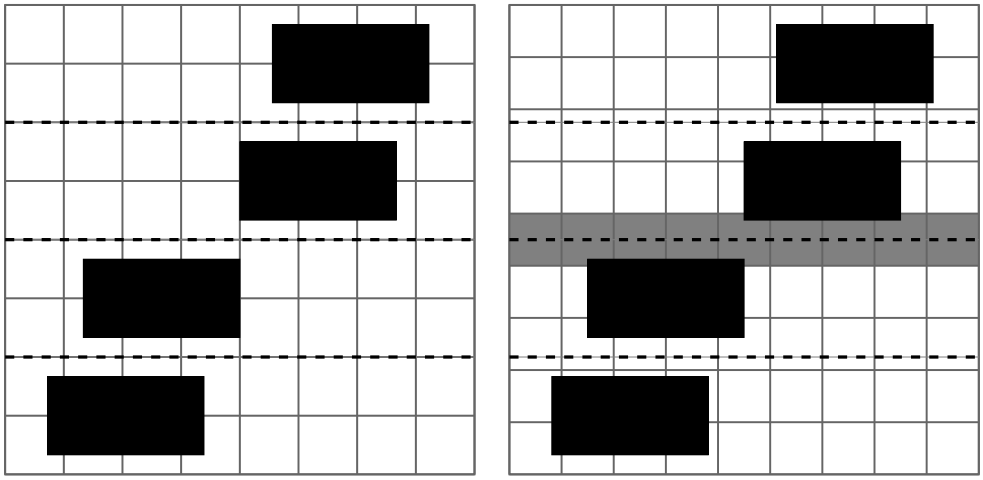
It is difficult to get a fair comparison of traffic signals and FCFS in a single-intersection scenario. FCFS performs well with vehicles arriving in a Poisson fashion, whereas traffic signals perform poorly under these conditions. If the traffic coming into a traffic signal–controlled intersection is properly shaped, traffic signals can incur very low delays (the so-called "green wave"). For this reason, the results in this section should not be considered the authoritative quantitative comparison between traffic signals and a multiagent mechanism, as the Poisson arrivals may be favoring the latter over the former.

7.3 Choosing Granularity

Of note in Figure 7.1(a) is the spike in delay for the granularity-ratio-1 FCFS policy. The system looks as though it is behaving chaotically—in Figure 7.1(b), delay slowly and steadily increases with the traffic level, until spiking off the graph when the probability of spawning a vehicle each time step reaches about 0.013.

With the granularity-ratio-1 system, vehicles traveling parallel to one another compete for the same tiles. This situation also arises for vehicles in the lanes closest to the middle of the road whenever the granularity ratio is a small, odd number, as in Figure 7.2(b). Recall that in the prototype simulator, acceleration in the intersection is forbidden. Thus, if a vehicle slows down because it cannot obtain a reservation, when it finally does get a reservation it will be moving slowly for the entirety of the reservation and occupy the reservation tiles for a longer period of time. The next car to approach the intersection is therefore more likely to slow down as well. This process feeds itself and the vehicles slow down more and more. For small to average amounts of traffic, delays increase, but the system recovers during probabilistically generated periods of light traffic. However, for very heavy traffic, the intersection will eventually reach a deadlocked state. Because traffic is generated stochastically,

this could happen early or late in the experiment. If it happens early, it will have a large effect on the average delay, whereas if it happens late, the effect will be smaller. Deadlocking is difficult to measure quantitatively, because as it progresses, driver agents make reservations for very long periods of time—so long, in fact, that they overflow the memory of the computer running the simulator. This effect can be seen in the rough line in Figure 7.1(a). To further explore the effects of granularity, we ran several more experiments, varying the granularity as well as the number of lanes.



(a) Granularity ratio 8 (b) Granularity ratio 9

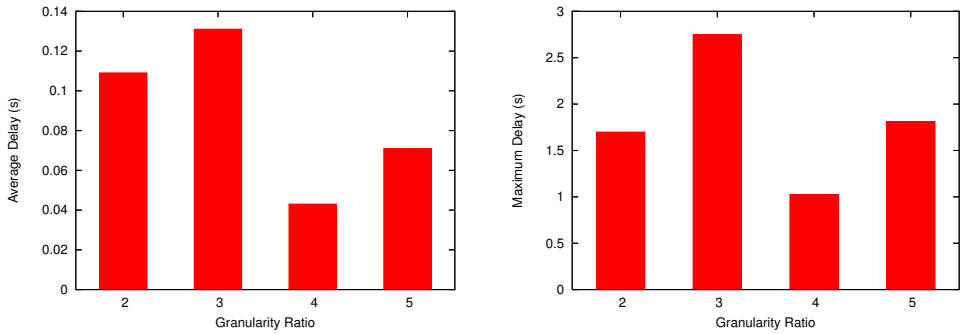
Figure 7.2: Increasing the granularity ratio does not always improve performance. In 7.2(a), a granularity ratio of 8 suffices. In 7.2(b), increasing the granularity ratio to 9 actually hurts performance—vehicles traveling parallel to each other (but in opposite directions) are competing for the middle row of tiles.

7.3.1 Experimental Setup

These experiments were also performed in the `aim1` simulator as described in Section 7.2.1. Each data point represents 500,000 steps of simulation (approximately 2.5 hours of simulated time). The traffic level is fixed at 0.2 vehicles per second.

7.3.2 Results

As shown in Figure 7.3, with 2 lanes in each direction, a 2×2 grid performs better than a 3×3 grid. Increasing to a 4×4 grid is better than 2×2, but increasing it to 5×5 is again worse. An increase in granularity ratio should correspond to a decrease in delay. However, for small granularity ratios, incrementing the granularity ratio from a small even number to a small odd number actually *increases* delay. In the case of maximum delay, even the granularity-ratio-2 system performs better than the granularity-ratio-5 system; the ill effects of odd granularity ratios as shown in Figure 7.2 tend to slow down a few unfortunate vehicles.



(a) Average delay (b) Maximum delay

Figure 7.3: Simulation statistics for FCFS policies with varying granularity. There are 2 lanes in each direction and the traffic level is 0.2 vehicles per second. Each experiment is run for 500,000 simulation steps. Note that increasing the granularity does not always improve performance.

This experiment suggests that FCFS should always be run with granularity ratio high enough such that vehicles that never cross paths never compete for the same reservation tiles. As Figure 7.4 shows, more lanes require a higher granularity ratio (though even with low granularity ratio, the system out-performs the traffic signal). However, because the computational complexity of the system increases proportional to the square of the granularity ratio, the granularity ratio should not be increased indiscriminately.

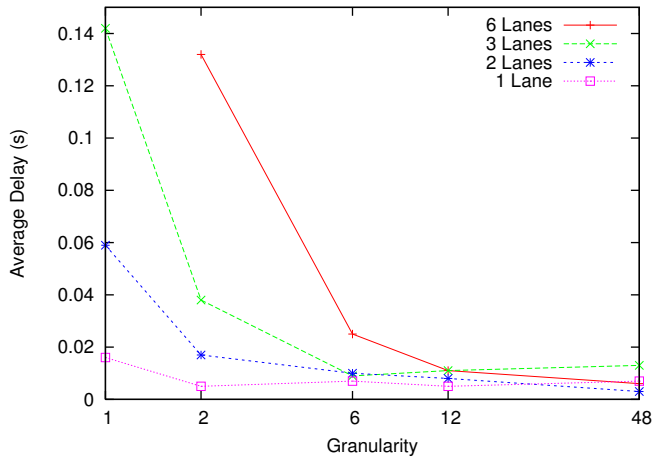


Figure 7.4: Average delays for FCFS with independently varying numbers of lanes and granularity ratio. Increasing the granularity ratio beyond twice the number of lanes results in only marginal improvements. All simulations were run for at least 500,000 steps. 6 lanes with 1 tile deadlocks and overflows the system memory before 500,000 steps can complete.

7.4 The Full Power of FCFS

While earlier experiments used the `aim1` simulator, these experiments use the full power of FCFS, including turning and acceleration, which were first made possible in `aim2`. Because vehicles turn, and thus do not always travel within a line of reservation tiles, increasing granularity beyond twice the number of lanes can improve performance even more. In addition to FCFS, we evaluate the stop sign policy as presented in Chapter 4.2.

In Chapter 6, I described how the `aim3` simulator measures delay on a time-step by time-step basis. This capability was not present in `aim2`, where vehicle delays were not adjusted to account for the fact that vehicles must slow to make turns. So while technically, the optimal delay for an individual vehicle is no delay at all, in `aim2`, a vehicle could record a small amount of delay just from the need to slow to avoid losing control. In order to create a worthwhile benchmark against

which to compare the reservation system, we first empirically measured the optimal *average* delay for an intersection manager. To make this measurement, we created a special control policy that accepts all requests. We also deactivate each vehicle's ability to detect other vehicles, eliminating the interactions between them. These results are presented as the "optimal" control policy, which while optimal in terms of non-adjusted delay, provides no safety guarantees.

Small intersections with slow-moving traffic tend not to be amenable to control by traffic signals. Very light traffic can usually regulate itself fairly effectively. For example, consider an intersection with a stop sign—all vehicles must come to a stop, but afterwards may proceed if the intersection is clear. In these situations, a stop sign is often much more efficient than a traffic signal, because vehicles are never stuck waiting for a signal to change when there is no cross-traffic. The protocol enables us to define such a control policy, and we compare it experimentally to the other policies. Note that this policy is much more efficient than an actual stop sign, because once the vehicle has stopped at the intersection, the driver agent and intersection can determine when the car may safely proceed much more precisely and much less conservatively than a human driver.

7.4.1 Experimental Setup

The simulator simulates 3 lanes in each of the 4 cardinal directions. The speed limit in all lanes is 25 meters per second. Every configuration shown is run for at least 100,000 steps in the simulator, which corresponds to approximately half an hour of simulated time. Vehicles that are spawned turn with probability 0.1, and turning vehicles turn left or right with equal probability. Vehicles turning right are spawned in the right lane, whereas vehicles turning left are spawned in the left lane. Vehicles that are not turning are distributed probabilistically amongst the lanes such that the traffic in each lane is as equal as possible. FCFS and the stop sign (implemented as an extension of FCFS—see Chapter 4.2) both have a granularity ratio of 24.

7.4.2 Results

The results for the experiments are shown in Figure 7.5. As expected, the average delay for the optimal system is positive and nonzero, but very small.

FCFS performs very well, nearly matching the performance of the optimal policy. At higher levels of traffic, the average delay for a vehicle gets as high as 0.35 seconds, but is never more than 1 second above optimal. Under none of the tested conditions does FCFS even approach the delay of the traffic signal system from the previous experiment, shown in Figure 7.1(a).

The stop sign does not perform as well as FCFS, but for low amounts of traffic, it still performs fairly well, with average delay only about 3 seconds greater than optimal. However, as the traffic level increases, performance degrades. It is difficult to imagine a scenario in which this implementation of the stop sign would actually be used—it requires the same technology as the reservation system, but does not have any advantages over FCFS—it is presented here only as an approximation of an actual stop sign.

7.5 Allowing Turns from Any Lane

In traditional traffic systems, especially those with traffic signals, vehicles wishing to turn onto the cross street must do so from specially designated turning lanes. This extra lane prevents cars that want to turn from holding up non-turning traffic. However, with a system like the reservation system, such a specialized lane is no longer necessary. There is nothing inherent in the reservation system that demands vehicles turn from any specific lane. Investigating the effects of allowing turning from any lane produced some surprising results. As seen in Figure 7.6, relaxing the restriction actually hurts FCFS's performance slightly. While one might think this allows the vehicles more flexibility, on average it increases the resources used by any one turning vehicle. By making left turns from the left lane and right turns

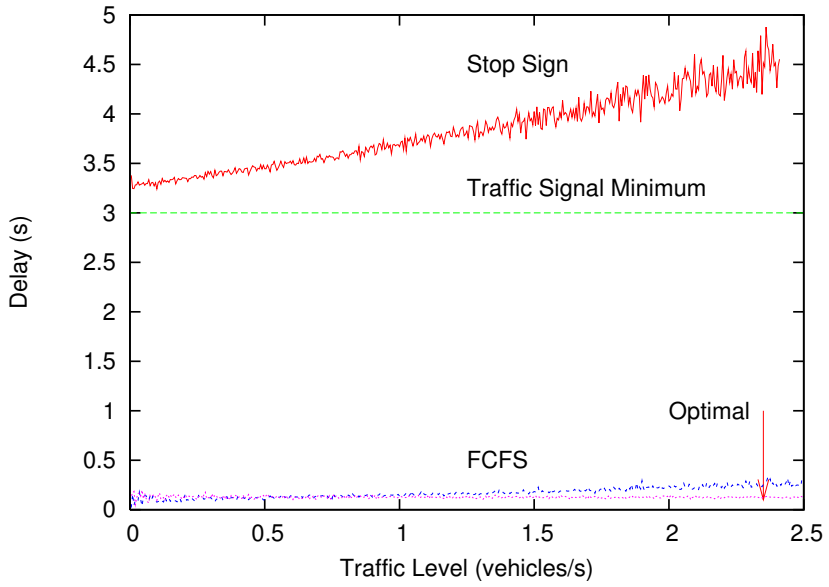


Figure 7.5: Delays for varying amounts of traffic for FCFS, the stop sign, and the optimal system.

from the right lane, vehicles both travel a shorter distance and reserve reservation tiles that are less heavily used. However, these experiments may be misleading. Vehicles changing lanes to get into a designated turn lane could potentially delay vehicles behind them in the process. Because have not yet developed a robust lane-changing behavior in `aim3` (which does model lane changing), we have not been able to experimentally verify this conjecture.

7.6 Emergency Vehicle Experiments

While we have already shown that FCFS on its own can significantly reduce average delays for all vehicles, FCFS-Emerg helps reduce delays for emergency vehicles even further.

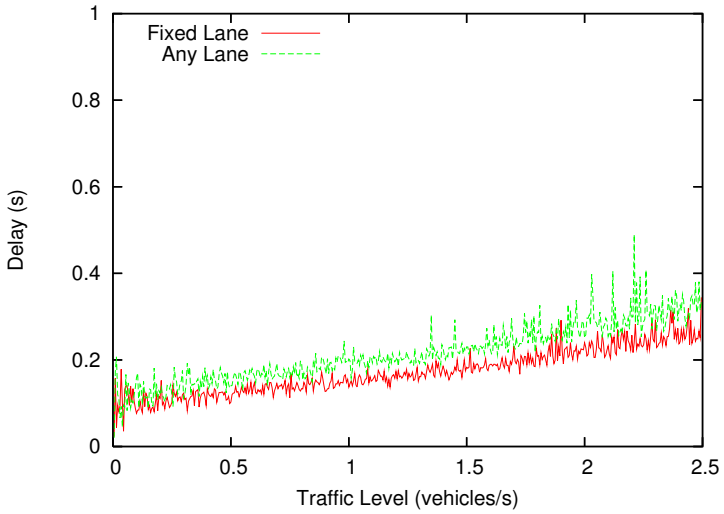


Figure 7.6: Comparison of an FCFS policy with traditional turns to one allowing turning from any lane. Allowing turns from any lane decreases performance slightly, producing longer delays.

7.6.1 Experimental Setup

To demonstrate this improvement, we ran the simulator with varying amounts of traffic, while keeping the proportion of emergency vehicles fixed at 0.1% (that is, a spawned vehicle is made into an emergency vehicle with probability 0.001). Because of the very small number of emergency vehicles created with realistically low proportions, we ran each configuration (data point) for 100 hours of simulated time—much longer than the other experiments.

7.6.2 Results

As shown in Figure 7.7, the emergency vehicles on average experience lower delays than the normal vehicles. The amount by which the emergency vehicles outperform the normal vehicles increases as the traffic increases, suggesting that as designed, FCFS-Emerg helps most when more traffic is contending for space-time in the intersection.

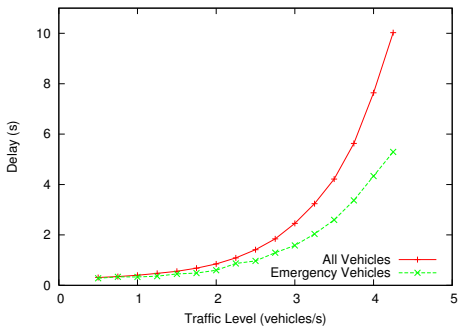
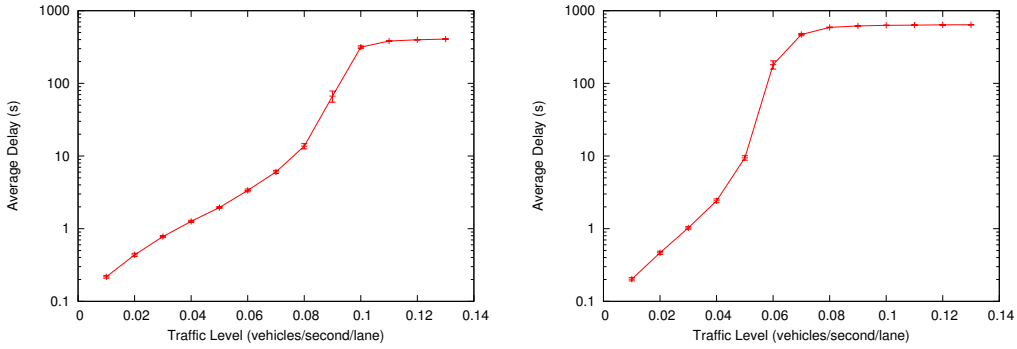


Figure 7.7: Average delays for all vehicles and emergency vehicles as a function of traffic level for the FCFS-EMERG policy. One out of a thousand vehicles (on average) is an emergency vehicle. Delays for the emergency vehicles are lower for all data points.

7.7 V2V Performance

Figure 7.8 shows results from our V2V experiments. In 7.8(a), a single lane enters and exits the intersection in each direction. At around 0.08 vehicles per second per lane, the intersection reaches maximum throughput. Above 0.1 vehicles per second per lane, the simulator becomes saturated and cannot increase the amount of traffic, so delay levels off. At lower traffic levels, the delay is quite low. In 7.8(b), the same experiment is run, but with two lanes entering and exiting the intersection in each direction. This time, the intersection reaches capacity at a lower traffic level per lane. However, because there are more lanes, the overall traffic level is about the same (the intersection has about the same throughput). While this might seem counterintuitive, consider the fact that the intersection is larger, and each trajectory has many more incompatible trajectories. A vehicle crossing this intersection will spend more time in the intersection, and block more lanes of traffic while doing so. It thus stands to reason that on a per-lane basis, the maximum throughput will decrease. This odd relationship between intersection size and throughput further reinforces the restriction of the V2V system to small, low-traffic intersections. These experiments were carried out in the latest version of the simulator, `aim3`, with equal

parts coupe, sedan, sport/utility vehicle (SUV), and van (see Chapter 6.3.6 for descriptions).



(a) One lane

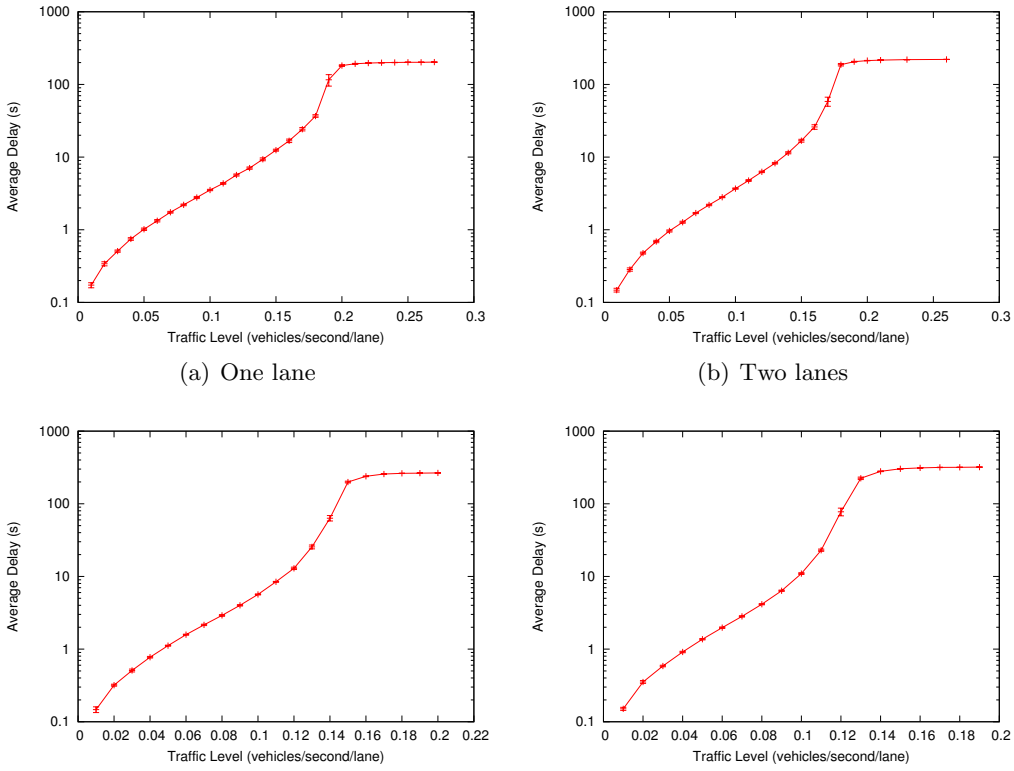
(b) Two lanes

Figure 7.8: Average delay for intersections using the V2V mechanism. The y-axis is a log scale. The 2-lane intersection can handle less traffic per-lane, but can handle about the same amount of total traffic.

7.8 Pushing λ In FCFS

The previous section showed results for the V2V system as the traffic per lane, λ, was increased. The results also gave a good indication of the maximum throughput under the V2V mechanism. In this section, we perform a similar experiment with FCFS. By increasing λ until the maximum throughput is reached, we can determine what that maximum throughput is. Keep in mind that this limit is a function of more than just the type of policy at work in the intersection manager. The size of the various buffers also has an effect on how efficient the intersection can be. In all of these experiments, the static buffer size was 0.5 meters, the time buffer on internal tiles was 0.25 seconds, and the edge tile time buffer was 2 seconds. Decreasing these buffers leads to higher throughput, while increasing them leads to lower throughput. As these buffer settings are somewhat arbitrary, and overall performance is a function of many more constants in the simulator, the throughput numbers should not be considered directly comparable to real-world numbers.

Figure 7.9 shows the results from this experiment. In 7.9(a), a single lane travels in each direction. As the traffic per lane approaches 0.18 vehicles per second, the intersection's maximum throughput is reached. The simulator quickly runs out of resources to simulate enough vehicles to sustain the growth in delay, which levels out around 0.2 vehicles per second per lane. For 7.9(b), 7.9(c), and 7.9(d), each of these transitions happens at a lower traffic level, but the overall throughput of the intersection still increases, as the effect is not as dramatic as it is with a V2V intersection. The FCFS policy makes more efficient use of the space-time in the intersection. These experiments were also carried out in the latest version of the simulator, `aim3`, with equal parts coupe, sedan, sport/utility vehicle (SUV), and van (see Chapter 6.3.6 for descriptions).



(a) One lane

(b) Two lanes

(c) Three lanes

(d) Four lanes

Figure 7.9: Pushing traffic to the maximum level for FCFS-controlled intersections with 1, 2, 3, and 4 lanes traveling in each direction. The y-axis is a log scale. As with Figure 7.8, throughput per lane decreases as the number of lanes increases, but not as dramatically. Overall throughput still increases as more lanes are added.

Chapter 8

Human Usability

While an intersection control mechanism for autonomous vehicles will someday be very useful, there will always be people who enjoy driving. Additionally, there will be a fairly long transitional period between the current situation (all human drivers) and one in which human drivers are a rarity. Even if switching to a system comprised solely of autonomous vehicles were possible, pedestrians and cyclists must also be able to traverse intersections in a controlled and safe manner. For this reason, it is necessary to create intersection control policies that are aware of and able to accommodate humans, whether they are on a bicycle, walking to the corner store, or driving a "classic" car for entertainment purposes. In this section we explain how we have extended the FCFS policy and the reservation framework to incorporate human drivers. In order to accommodate human drivers, a control policy must be able to direct both human and autonomous vehicles, while coordinating them, despite having much less control and information regarding where and when the human drivers will be. The main concept behind our extension is the assumption that there is a human-driven vehicle anywhere one *could* be. While this may be less efficient than an approach which attempts to more precisely model human behavior, it is guaranteed to be safe, one of the desiderata on which we are unwilling to compromise. Adding pedestrians and cyclists follows naturally, and we give brief

descriptions of how this would differ from the extensions for human drivers.

Compatibility with human drivers offers more than the ability to handle the occasional human driver once the levels of human drivers in everyday traffic reaches a steady state. It will also help facilitate the transition from the current standard— all human-driven vehicles — to this steady state, in which human drivers are scarce. In Chapter 2.1, we emphasized the need for incremental deployability. As we will show experimentally, human compatibility adds significantly to the incremental deployability of the reservation system. We will also show that the specifics of the implementation offer further benefits: incentives for both communities and private individuals to adopt autonomous vehicle technology.

8.1 Using Existing Infrastructure

A reliable method of communicating with human drivers is a prerequisite for including them in the system. The simplest and best solution is to use something human drivers already know and understand — traffic signals. Traffic signal infrastructure is already present at many intersections and the engineering and manufacturing of traffic signal systems is well developed. For pedestrians and cyclists, standard "push-button" crossing signals can be used that give enough time for a person to traverse the intersection. These can also serve to alert the intersection to their presence.

8.1.1 Signal Models

If real traffic signals are to be used to communicate to human drivers, they must be controlled and understood by the intersection manager. Thus, we add a new component to each intersection control policy, called a *signal model*. The signal model controls the physical signals as well as providing information to the policy with which it can make decisions. In more complicated scenarios, the signal model can be modified by the control policy, for example, in order to adapt to changing

traffic conditions. The signals have the same semantics as modern-day signals: red (do not enter), yellow (if possible, do not enter; signal will soon be red), and green (enter). Each control policy requires a signal model so that human users know what to do. For instance, the signal model for FCFS keeps all the signals red at all times, indicating to humans that it is never safe to enter. The Traffic-Light policy's signal model, on the other hand, corresponds exactly to the signal system the policy is emulating. Here, we describe a few signal models used in our experiments.

All-Lanes

In this model, which is very similar to some current traffic signal systems, each direction in succession gets green signals in all lanes. Thus, all northbound traffic (turning and going straight) has green signals while the eastbound, westbound, and southbound traffic all have red signals. The green signals then cycle through the directions. As it is similar to some current traffic signals, this signal model is particularly well-suited to controlling distributions of vehicles with significant contingents of human drivers. We demonstrate this fact in our experimental results. Figure 8.1 shows a graphical depiction of this signal model. Videos of the `aim2` simulator running the FCFS-Signal policy with the All-Lanes signal model can be seen on the videos section of the project page at `http://www.cs.utexas.edu/~kdresner/aim/`.

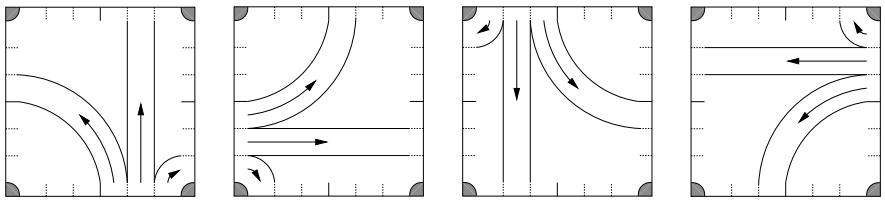


Figure 8.1: The All-Lanes signal model. Each direction gets all green signals in a cycle: north, east, south, west. During each phase, the only available paths for autonomous vehicles with red signals are right turns.

Single-Lane

In the SINGLE-LANE signal model, the green signal rotates through the lanes one at a time instead of by direction. For example, the left turn lane of the northbound traffic would have a green signal, while all other lanes would have a red signal. Next, the straight lane of the northbound traffic would have a green signal, then the right turn. Next, the green signal would go through each lane of eastbound traffic, and so forth. A graphical description of the model's cycle can be seen in Figure 8.2. This signal model does not work very well if most of the vehicles are human-driven, but as we will show, is very useful for intersections which control mostly autonomous vehicles but need also to handle an occasional human driver. Videos of the `aim2` simulator running the FCFS-SIGNAL policy with the SINGLE-LANE signal model can be seen on the videos section of the project page at `http://www.cs.utexas.edu/~kdresner/aim/`.

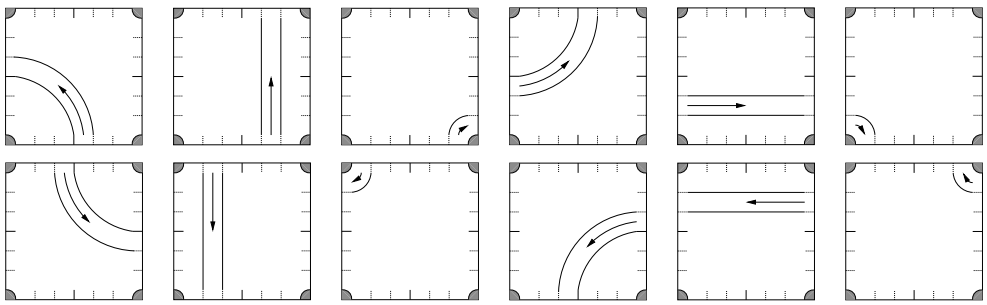


Figure 8.2: The SINGLE-LANE signal model. Each individual lane gets a green signal (left turn, straight, then right turn), and this process is repeated for each direction. Note how a smaller part of the intersection is used by human vehicles at any given time. The rest of the intersection is available to autonomous vehicles.

8.2 The FCFS-Signal Policy

In order to obtain some of the benefits of the FCFS policy while still accommodating human drivers, a policy needs to do two things:

1. If a signal is green, ensure that it is safe for any vehicle (autonomous or human-driven) to drive through the intersection in the lane the signal regulates.

2. Grant reservations to driver agents whenever possible. Autonomous vehicles can thus move through red signals (whereas humans cannot), provided they have a reservation—similar to a "right on red", but extended much further to other safe situations.

The policy FCFS-SIGNAL, which does both of these, is described as follows:

- As with FCFS, the intersection is divided into a grid of $n \times n$ tiles.
- Upon receiving a request message, the policy uses the parameters in the message to establish when the vehicle will arrive at the intersection.
- If the signal controlling the lane in which the vehicle will arrive at the intersection will be green at that time, the reservation is confirmed.
- If the signal controlling the lane will be yellow, the reservation is rejected.
- If the signal controlling the lane will be red, the journey of the vehicle is simulated as in FCFS.
- If throughout the simulation, no required tile is reserved by another vehicle or in use by a lane with a green or yellow signal, the policy reserves the tiles and confirms the reservation. Otherwise, the request is rejected.

8.2.1 Off-Limits Tiles

Unfortunately, simply deferring to FCFS does not guarantee the safety of the vehicle. If the vehicle were granted a reservation that conflicts with a vehicle following the physical signals, a collision could easily ensue. To determine which tiles are in use by the signal system at any given time, we associate a set of *off-limits tiles* with

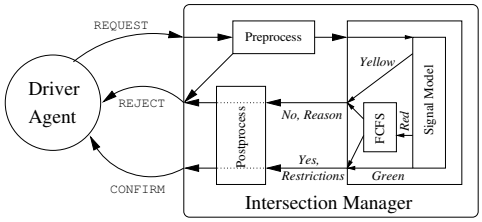


Figure 8.3: FCFS-SIGNAL is the combination of FCFS and a signal model. When a request is received, FCFS-SIGNAL first checks to see what color the signal will be. If it is green, it grants the request. If it is yellow, it rejects. If it is red, it defers to FCFS.

each signal. For example, if the signal for the northbound left turn lane is green (or yellow), all tiles that could be used by a vehicle turning left from that lane are considered reserved for the purposes of FCFS. The length of the yellow signal is adjusted so that vehicles entering the intersection have enough time to clear the intersection before those tiles are no longer off limits.

8.2.2 FCFS-Signal Subsumes FCFS

Using a traffic signal–like signal model (for example ALL-LANES), FCFS-SIGNAL can behave exactly like TRAFFIC-LIGHT if all drivers are human. With a signal model that keeps all signals constantly red, FCFS-SIGNAL behaves exactly like FCFS. In this case, if any human drivers are present it will fail spectacularly, leaving the humans stuck at the intersection indefinitely. However, in the absence of human drivers, it will perform exceptionally well. FCFS is just a special case of FCFS-SIGNAL. We can thus alter FCFS-SIGNAL's behavior to vary from strictly superior to TRAFFIC-LIGHT to exactly that of FCFS.

8.3 Human Usability Experiments

In Chapter 7.4, we showed that once all vehicles are autonomous, intersection-associated delays can be reduced dramatically. The following experiments suggest

a stronger result: by using the two signal models presented in this Chapter, delays can be reduced at each stage of adoption. Furthermore, additional incentives exist at each stage for drivers to switch to autonomous vehicles.

8.3.1 Experimental Setup

For these experiments, we used the `aim2` simulator, which models 3 lanes in each of the 4 cardinal directions. The speed limit in all lanes is 25 meters per second. For each intersection control policy with reservation tiles, the granularity is 24. The simulator spawns all vehicles turning left in the left lane, all vehicles turning right in the right lane, and all vehicles traveling straight in the center lane[1]. Unless otherwise specified, each data point represents 180000 time steps, or one hour of simulated time. Our simulated human-driven vehicles use a two-second following distance, but use the same lane-following algorithm as the autonomous drivers. They also employ a "point-of-no-return" mechanism for reacting to signals—if the vehicle can stop at a yellow or red signal, it does, otherwise it proceeds.

8.3.2 Results

We present the experimental results for the human-compatible policies in two parts. The first focuses on how these policies can facilitate a smooth transition to an all-autonomous or mostly-autonomous vehicle system. The second focuses on the incentives throughout this process, both global and individual, to continue deployment of the system. Combined, these results suggest that an incremental deployment (one of the desiderata) is both technically possible and desirable.

[1]This is a constraint we will likely relax in the future. It is included in this work to give the Single-Lane signal model more flexibility and for a fair comparison to the FCFS policy, which performs even better in its absence.

Transition To Full Deployment

The purpose of a hybrid intersection control policy is to confer the benefits of autonomy to passengers with driver-agent controlled vehicles while still allowing human users to participate in the system. Figure 8.4 shows a smooth and monotonically improving transition from modern-day traffic signals (represented by the Traffic-Light policy) to a completely or mostly autonomous vehicle mechanism (FCFS-Signal with the Single-Lane signal model). In early stages (100%-10% human), the All-Lanes signal model is used. Later on (less than 10% human), the Single-Lane signal model is introduced. At each change (both in driver populations and signal models), delays are decreased. Notice the rather drastic drop in delay from FCFS-Signal with the All-Lanes signal model to FCFS-Signal with the Single-Lane signal model. Although none of the results is quite as close to the minimum as pure FCFS, the Single-Lane signal model allows for greater use of the intersection by the FCFS portion of the FCFS-Signal policy, which translates to higher efficiency and lower delay.

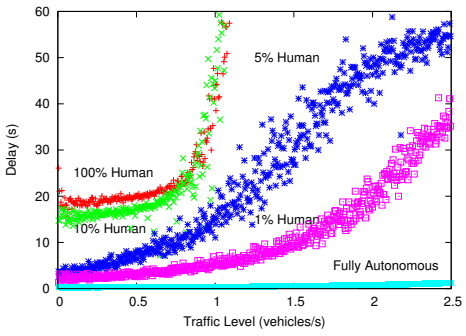


Figure 8.4: Average delays for all vehicles as a function of traffic level for FCFS-Signal with two different signal models: the All-Lanes signal model, which is well-suited to high percentages of human-driven vehicles, and the Single-Lane signal model, which only works well with relatively few human-driven vehicles. As adoption of autonomous vehicles increases, average delays decrease.

For systems with a significant proportion of human drivers, the All-Lanes

signal model works well—human drivers have the same experience they would with the Traffic-Light policy, but autonomous driver agents have extra opportunities to make it through the intersection. A small amount of this benefit is passed on to the human drivers, who may find themselves closer to the front of the lane while waiting for a red signal to turn green. To explore how much the average vehicle would benefit, we ran our simulator with the FCFS-Signal policy, the All-Lanes signal model, and a 100%, 50%, and 10% rate of human drivers. This means that when a vehicle is spawned, it receives a human driver (instead of a driver agent) with probability 1, .5, and .1 respectively. As seen in Figure 8.5, as the proportion of human drivers decreases, the delay experienced by the average driver also decreases. While these decreases are not as large as those brought about by the Single-Lane signal model, they are at least possible with significant numbers of human drivers.

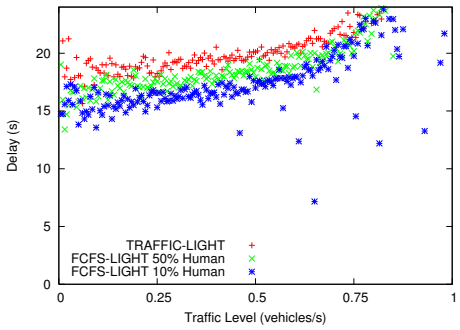


Figure 8.5: Average delays for all vehicles as a function of traffic level for FCFS-Signal with the All-Lanes signal model. Shown are the results for 100%, 50%, and 10% human-driven vehicles. The 100% case is equivalent to the Traffic-Light policy. Note that the average delay decreases as the percentage of human-driven vehicles decreases.

Incentives For Individuals

Even without any sort of autonomous intersection control mechanism, there are incentives for humans to switch to autonomous vehicles. Not having to do the driving, as well as the myriad safety benefits are strong incentives to promote autonomous

vehicles in the marketplace. Our experimental results suggest additional incentives. Using our reservation system, autonomous vehicles experience lower average delays than human-driven vehicles and this difference increases as autonomous vehicles become more prevalent.

Figure 8.6 shows the average delays for human drivers as compared to autonomous driver agents for the FCFS-SIGNAL policy using the ALL-LANES signal model. In this experiment, half of the drivers are human. Humans experience slightly longer delays than autonomous vehicles, but not worse than with the TRAFFIC-LIGHT policy (see Chapter 4.2). Thus, by putting some autonomous vehicles on the road, all drivers experience equal or smaller delays as compared to the current situation. This result is expected because the autonomous driver can do everything the human driver does and more.

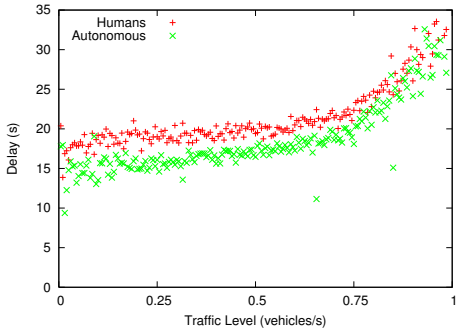


Figure 8.6: Average delays for human-driven vehicles and all vehicles as a function of traffic level for FCFS-SIGNAL with the ALL-LANES signal model. In this experiment, 50% of vehicles are human driven. Autonomous vehicles experience slightly lower delays across the board, and human drivers experience delays no worse than the TRAFFIC-LIGHT policy.

Once the reservation system is in widespread use and autonomous vehicles make up a vast majority of those on the road, the door is opened to an even more efficient signal model for the FCFS-SIGNAL policy. With a very low concentration of human drivers, the SINGLE-LANE signal model can drastically reduce delays, even

at levels of overall traffic that the TRAFFIC-LIGHT policy cannot handle. Using this signal model, autonomous drivers can pass through red signals even more frequently because fewer tiles are off-limits at any given time. In Figure 8.7 we compare the delays experienced by autonomous drivers to those of human drivers when only 5% of drivers are human and thus the SINGLE-LANE signal model can be used. While the improvements using the ALL-LANES signal model benefit all drivers to some extent, the SINGLE-LANE signal model's sharp decrease in average delays (Figure 8.4) comes at a high price to human drivers.

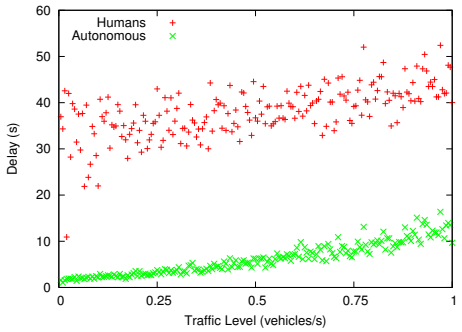


Figure 8.7: Average delays for human-driven vehicles and all vehicles as a function of traffic level for FCFS-SIGNAL with the SINGLE-LANE signal model. Humans experience worse delay than with TRAFFIC-LIGHT, but average delay for all vehicles is much lower. In this experiment, 5% of vehicles are human-driven.

As shown in Figure 8.7, human drivers experience much higher delays than average. For lower traffic levels, these delays are even higher than those associated with the TRAFFIC-LIGHT policy. Figure 8.4 shows that at high levels of traffic, human drivers benefit relative to TRAFFIC-LIGHT. Additionally, intersections using FCFS-SIGNAL will still be able to handle far more traffic than those using TRAFFIC-LIGHT.

The SINGLE–LANE signal model effectively gives the humans a high, but fairly constant delay. Because the green signal for any one lane only comes around after each other lane has had a green signal, a human-driven vehicle may find itself

sitting at a red signal for some time before the signal changes. However, since this signal model would only be put in operation once human drivers are fairly scarce, the huge benefit to the other 95% or 99% of vehicles far outweighs this cost. A signal model that detects and reacts to the presence of human drivers might be able to achieve even better overall performance, without causing the human drivers to wait as long.

These data suggest that there will be an incentive to both early adopters (persons purchasing vehicles capable of interacting with the reservation system) and to cities or towns. Those with properly equipped vehicles will get where they are going faster (not to mention more safely). Cities and towns that equip their intersections to utilize the reservation paradigm will experience fewer traffic jams and more efficient use of the roadways (along with fewer collisions and less wasted gasoline). Because there is no penalty to the human drivers (which would presumably be a majority at this point), there would be no reason for any party involved to oppose the introduction of such a system. Later, when most drivers have made the transition to autonomous vehicles, and the SINGLE-LANE signal model is introduced, the incentive to move to the new technology is increased—both for cities and individuals. By this time, autonomous vehicle owners will far outnumber human drivers, who will still benefit when traffic is at its worst.

8.4 Automatic Switching and Policy Selection

A major benefit of the switching mechanism explained in Chapter 4.3 is that the intersection manager need not always choose a policy capable of handling the maximum possible proportion of human drivers. Instead, as traffic conditions change, the manager can adjust the policy to compensate, increasing efficiency during periods in which human drivers are more scarce. In this section, I discuss a method for enabling the intersection manager to, completely autonomously, select and switch

to a new policy based on the current traffic conditions. Furthermore, this method does not require the addition of any new sensing infrastructure or communication from driver agents—it operates entirely by analyzing existing communication.

8.4.1 The Cost Of Switching

During a switch between P and P', the mechanism insists that no vehicle can enter the intersection before $last_P$ unless it also exits before $last_P$. For a brief instant at time $last_P$, there can be no vehicles in the intersection; there is a "wall" (in time) that cannot be crossed. Autonomous vehicles that would otherwise be in the intersection at $last_P$ must accelerate or decelerate such that they get a reservation which will be completed before $last_P$ or begin after $last_P$. Placing additional constraints on the vehicles could decrease the overall efficiency of the intersection, increasing delays. This would create an interesting tradeoff: switching policies could have a benefit, but it might not outweigh the cost of making the switch.

However, we determined that with the FCFS-Signal policy, no real tradeoff exists. FCFS-Signal's off-limits tiles already create many "walls" (in space) that cannot be traversed, and the addition of the constraint made by switching does not have a significant effect. To quantify the effects of switching, we ran a series of 24-hour simulations in which the intersection manager repeatedly "switched" from an FCFS-Signal policy with the Single-Lane signal model to an identical policy at regular intervals. In the experiment, we set the vehicle spawning probability to a moderate 0.01 — enough that vehicles would actually compete for passage through the intersection, but also low enough to make random congestion unlikely. The baseline time for a vehicle to complete its trip is 10 seconds — 250 meters at the speed limit of 25 m/s. By varying the time between switches from 24 hours (effectively ∞) to 5 seconds, we determined that the policy switching has no significant negative effects until the switches occur extremely frequently. At the highest frequencies, the "walls" created by the switch create compartments in space-time that are only

slightly longer than the time it takes a vehicle to traverse the intersection. At this point, it becomes more difficult for the intersection manager to fit a vehicle into the available space-time. Table 8.1 presents the results from this experiment.

Period	**Delay(s)**	**CI(95%)**
∞	2.03	$\pm$0.01
1h	2.03	$\pm$0.01
10m	2.03	$\pm$0.01
1m	2.13	$\pm$0.01
30s	2.20	$\pm$0.01
10s	4.25	$\pm$0.1
5s	5.14	$\pm$0.07

Table 8.1: The policy switching mechanism has no effect on delay until the time between switches approaches the time it takes to traverse the intersection.

8.4.2 Policy Selection

The two signal models described earlier, All-Lanes and Single-Lane, each define a different intersection control policy when combined with FCFS-Signal. All-Lanes is suited to scenarios involving many humans, while Single-Lane is better for scenarios in which humans are scarce. Determining which policy to use should thus be as simple as determining how many of the vehicles using the intersection are not autonomous. Unfortunately, a direct approach would involve additional expensive infrastructure, either sensors at the intersection or signaling devices on the human-driven vehicles. The humans drivers may not even be willing to place such signaling devices on their vehicles due to privacy concerns.

Instead, we base our choices on the information already available to the intersection manager via the reservation requests made by the autonomous vehicles. If an autonomous vehicle is stuck behind a human vehicle waiting at a red signal, the parameters of the autonomous vehicle's next reservation request will change. It may even be forced to cancel. One altered message may not contain much information,

but the intersection manager communicates with many vehicles. By maintaining a sliding window of statistics from these messages, we can gather enough information about the current state of traffic such that a trained classifier can select the most appropriate policy. Our first instinct was to use a regression learner to estimate the average delay under the various candidate policies, allowing the intersection manager to choose the policy with the lowest estimate, but the regression learner proved unreliable. Instead, we learn the choice the intersection manager must make: which policy to use.

The classifier has 7 inputs:

- The current policy
- The rate (requests/second) at which the intersection manager is receiving reservation requests
- The rate (cancelations/second) at which the intersection manager is receiving reservation cancelations
- The rate (changes/second) at which the intersection manager is receiving reservation change requests
- The average time before the start of a reservation that requests are made
- The average velocity at which autonomous vehicles expect to arrive at the intersection
- The ratio of accepted reservations to total requests

8.4.3 Generating Training Data

We created a large body of training data by simulating over 800 one-hour episodes, half using All-Lanes and half using Single-Lane. Each episode included a five-minute "warm-up" period during which no data were recorded, to eliminate the ef-

fects of starting with an empty intersection. Classifier input data were then recorded in sliding windows from 2.5 to 30 minutes long. At the end of the episode, we set the target policy for each generated instance to the policy that had the lowest average delay at the end of the episode. Each episode used randomized traffic conditions, however every randomly-generated configuration was used twice — once for each policy. The spawning rate was chosen uniformly from the interval $(0.001, 0.025]$, which represents everything from very light to extremely heavy traffic. The proportion of human drivers was chosen uniformly from the interval $(0, 0.25]$. Above 25% humans, all but the lightest traffic scenarios favor All-Lanes.

8.4.4 Choosing a Classifier

With this data, we tested many different classifiers using the WEKA machine learning software [Witten and Frank, 2005]. We evaluated each classifier on each sliding window size with 10-fold cross-validation. Table 8.2 presents results from four representative classifiers: JRip (rules), J48 (decision tree), AdaBoost with decision stumps, and a neural network. Each classifier used WEKA's default settings.

	Classifier				
Window	Const.	AdaB.	J48	JRip	N.N.
2.5 min.	66.94	69.03	78.94	79.21	80.19
5 min.	66.95	71.15	80.33	81.23	82.19
10 min.	66.84	70.05	83.23	82.18	83.16
20 min.	65.64	74.10	81.66	81.44	84.88
30 min.	67.82	73.27	85.89	83.16	88.48

Table 8.2: Percentage of correctly classified instances on the training data using 10-fold cross-validation.

The neural network performed the best overall, followed closely by the J48 decision tree. All results were reported by WEKA as statistically significant (with respect to the constant classifier) with 95% confidence. As we had initially suspected, the longer sliding windows were much less noisy, and therefore easier to learn. For

the rest of the chapter, unless otherwise specified, when we refer to the classifier, we mean the neural network as implemented in WEKA and trained on the data from the 10-minute sliding windows. While exploring the space of potential training data might make for an interesting optimization, it is not the main focus of this work, and thus we fix this variable in order to study other aspects of the mechanism more closely. Because performance does not vary dramatically over the range tested, we chose a value near the middle of the range.

8.4.5 Putting the Classifier to Work

We combine the classifier with policy switching by maintaining a sliding window of data in the intersection manager, which the classifier uses to select a policy at pre-specified intervals. If the classifier chooses the policy already in use, no switch occurs. By integrating the trained classifier with the policy switching method, we produce an intersection manager capable of selecting a policy based on current traffic conditions, inasmuch as the traffic conditions are communicated through the reservation requests of autonomous vehicles. It is interesting to note that the classifier's target task and the simulations which generated its training data are subtly different. When generating training data, each policy was essentially in a steady state. However, in the target task the classifier must tolerate the fact that although current simulator settings may be best served with a particular policy, congestion created earlier in the experiment — perhaps the result of different simulator settings or a poor choice of policy — will affect the vehicles currently making reservation requests.

8.5 Policy Selection Experiments

To evaluate the performance of our classifier-based policy selection, we ran experiments in which the population of drivers was varied over time. The experiments pitted our autonomous intersection manager against the individual policies amongst

from which it could choose, as well as a custom policy-switching intersection manager with advance knowledge of which policy worked best for each driver population.

8.5.1 Experimental Setup

In these experiments, the `aim2` simulator models a 250m $\times$ 250m area with three lanes of traffic travel in each cardinal direction. Vehicles are limited to a maximum speed of 25m/s, and the granularity of each policy is 1/24 of the width and height of the intersection.

The test scenario comprises a series of 72 randomly generated simulator traffic settings. As with the training data generation, the spawning probability and human driver proportion were chosen uniformly at random from the intervals $(0.001, 0.025]$ and $(0, 0.25]$, respectively. Although the settings are randomly generated, we use the same sequence for each trial. Each trial lasts 72 simulated hours, with each configuration used for exactly one hour. Performance is measured by calculating the average delay over all vehicles spawned in the 72 hours. The switching managers use a sliding window to keep an average of all input values from the last time a decision was made; the size of the sliding window is equal to the time between potential switches.

8.5.2 A Lower Bound

After analyzing the performance of All-Lanes and Single-Lane throughout the 72-hour trial, we determined which policy worked best for each configuration and created an intersection manager that switches accordingly. We call this manager "omniscient", as it knows when to switch *a priori*. The omniscient manager is not technically optimal — it chooses the policy that performs best on each configuration, but it cannot adapt to changes in traffic conditions that result from the stochastic nature of the traffic generation.

8.5.3 Switch Frequency

While the configuration changes took place at regular intervals, a robust switching manager should not rely on such assumptions. By allowing switching more frequently, the intersection manager gains agility, but may pay a price in terms of stability. However, as we showed in Chapter 8.4.1, stability is not as important as one might think — the cost for making a policy switch is negligible. Agility turns out to be much more important: not only can the intersection manager react quickly to changing conditions, but it can also switch back quickly if it chooses the wrong policy. We created five versions of the intersection manager, varying the switching period from 20 minutes to 30 seconds. Note that whenever the manager selects the policy it is already using, no switch takes place. Table 8.3 shows the results of running each of these five versions, as well as ALL-LANES, SINGLE-LANE, and the omniscient intersection manager.

Policy		**Delay(s)**	**CI(95%)**
ALL-LANES		57.70	±0.43
SINGLE-LANE		48.30	±0.40
Switching	20m	43.28	±0.51
	10m	41.77	±0.46
	5m	41.53	±0.33
	1m	41.45	±0.66
	30s	41.05	±0.42
Omniscient		37.50	±0.45

Table 8.3: Average delay during a 72-hour simulated period. As the intersection manager switches policies more often, it can react to changing conditions more quickly, leading to lower average delay.

Every switching manager performed significantly better than either ALL-LANES or SINGLE-LANE alone. The switching manager with the shortest period (30 seconds) delayed the average vehicle only a few seconds more than the omniscient switcher. However, with the exception of the 20-minute version, the difference in performance between the learned switchers was not statistically significant. Per-

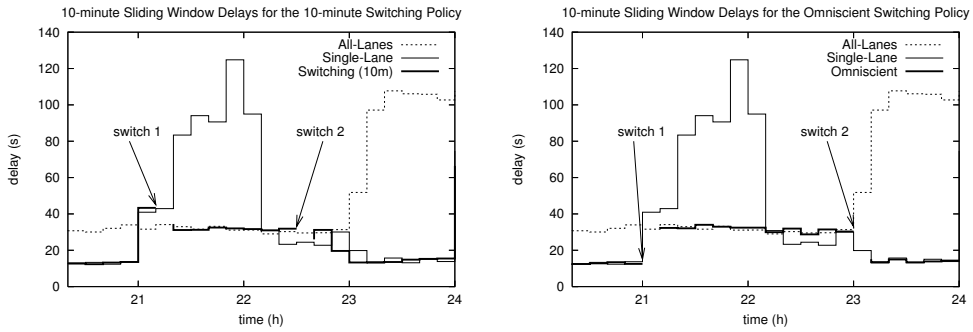
haps most importantly, re-evaluating the policy choice as frequently as every 30 seconds does not negatively affect performance — the classifier rarely recommends switches at successive 30-second decision points. These results do not indicate that SINGLE-LANE is better than ALL-LANES — it is trivial to adjust the traffic settings (specifically the proportion of human drivers) so that either policy performs better than the other. However, intelligent policy switching should always perform about as well or better than the best of the two static policies.

8.5.4 Outperforming The Omniscient Policy

The results in Table 8.3 show that the switching intersection manager can handle varying proportions of human drivers, even though it never directly senses or communicates with them — all information used by the classifier is readily available from the reservation requests of the autonomous vehicles. When we used the same 72 sets of simulator settings, but without any humans, the switching intersection manager behaved exactly as we expected: quickly switching to SINGLE-LANE and never going back. If the intersection manager were aware that the simulator was not spawning any human drivers, this would not be remarkable. However, the intersection manager gleans all its information about the current state of traffic from the reservation requests made by the autonomous vehicles. In some sense, the classifier-based switcher can outperform even the omniscient intersection manager at times, because the settings on the simulator do not precisely determine the actual traffic conditions — there is a lot of stochasticity.

Figure 8.8 shows the performance of the various policies on one representative section of the test scenario, with delay reported in 10-minute sliding windows. In 8.8(a), the classifier-based switcher, re-evaluating every 10 minutes, makes the switch from SINGLE-LANE to ALL-LANES as soon as it senses the change in traffic conditions. When conditions become more favorable to SINGLE-LANE, it makes a second switch back. However, the second switch is in the middle of the hour — it

does not correspond to a change in the simulator settings, but rather the actual traffic conditions. In contrast, Figure 8.8(b) shows the performance of the omniscient intersection manager on the same scenario. Notice that it makes the first switch preemptively, before the classifier-based manager would have any chance to sense a change. The traffic parameter change at 22 hours does not change the optimal policy, so the omniscient agent stays with All-Lanes. Because it is hard-coded to switch based on the actual simulator settings, it cannot sense the opportunity to switch later in that hour. Overall, however, the perfect prediction of the omniscient policy is more important than its inflexibility; the classifier does not always make the correct choice and this proves more significant (see Table 8.3).



(a) The classifier reacts to actual changes in traffic.

(b) The omniscient manager knows which policy is best overall for each configuration.

Figure 8.8: In 8.8(a), the classifier-based switcher first switches to All-Lanes once it senses traffic conditions have changed, then switches back to Single-Lane when conditions change the second time. In 8.8(b), the omniscient switcher knows in advance that conditions will change and preemptively switches to All-Lanes. However, because it is not adapting online, it does not switch back to Single-Lane until the traffic settings change at the end of the hour.

8.6 Summary

Traffic control is an inherently dynamic problem. Vehicles are moving, destinations are changing, and even the very makeup of the traffic varies, from hour to hour, day

to day, and year to year. In this chapter, I presented several mechanisms for dealing with this dynamicity on many different levels. The FCFS-Signal policy can take us from a world in which all vehicles are driven by humans, all the way to one in which all vehicles are autonomous. The two signal models that FCFS-Signal employs (as well as those models that have not yet been developed) can ensure that as more autonomous vehicles hit the roads, more of their capabilities are utilized to improve traffic conditions for all. The automated policy switching method allows intersection managers to respond to fluctuating conditions even on a localized level. While many of the most impressive benefits of autonomous intersection management manifest only when the vast majority of the vehicle population is autonomous, this chapter shows that even in the meantime, we can still use autonomous intersection management to effect significant improvements in our traffic control.

Chapter 9

Failure Mode Analysis

Fully autonomous vehicles promise enormous gains in safety, efficiency, and economy for transportation. However, before such gains can be realized, a plethora of safety and reliability concerns must be addressed. In the previous sections, we have assumed that all vehicles perform without gross malfunctions. In this chapter, we relax that assumption and demonstrate how our reservation-based mechanism reacts to scenarios in which such malfunctions occur. Additionally, we intentionally disable some elements of the system in order to investigate both their necessity and efficacy.

9.1 Causes of Accidents

A collision in purely autonomous traffic can have any number of causes, including software errors in the driver agent, a physical malfunction in the vehicle, or even meteorological phenomena. In modern-day traffic, such factors are largely ignored for two reasons. First, the exclusively human-populated system, with its generous margins for error, is not as sensitive to small or moderate aberrations. Second, none of these factors are significant with respect to driver error as causes of accidents [Wierwille *et al.*, 2002]. However, in the future of infallible autonomous driver agents, it is exactly these issues which will be the prevalent causes of au-

tomobile collisions. The safety allowances explained in Section 4.1.1 and 4.1.2 are adjustable—given some maximum allowable error in vehicle positioning, the buffers can be extended to handle that error—but no reasonable adjustment can account for gross mechanical malfunction like a blowout or failed brakes. Because these types of issues are infrequent, we believe the safety of the intersection control mechanism will be acceptable even if individual occurrences are slightly worse than accidents today.

9.2 Adding a Safety Net

One can easily imagine how badly an accident in such an efficient system could be without any reactive safety measures in place. Here, we explain how the system deals with these rare, but dangerous events. As we will show in Section 9.3, disabling the safety measures leaves the system prone to spectacular failure modes, sometimes involving dozens of vehicles. Intact, the measures make such events much more manageable.

9.2.1 Assumptions

In Section 9.3, we will show how our reactive safety measures can reduce the average number of vehicles involved in a crash from dozens to one or two. However, in order to employ these safety measures fully, we must make a few additional assumptions.

Detecting The Problem

First, we assume that the intersection manager is able to detect when something has gone wrong. While this is certainly a non-trivial assumption, without it, no substantial mitigation is possible. Simply put, the intersection manager cannot react to something it cannot detect. There are two basic ways by which the intersection manager could detect that a vehicle has encountered some sort of problem: the vehicle can inform the intersection manager, or the intersection manager can detect

the vehicle directly. For instance, in the event of a collision, a device similar to that which triggers an airbag can send a signal to the intersection manager. Devices like this already exist in aircraft to emit distress signals and locator beacons in the event of a crash. The intersection manager itself might notice a less severe problem, such as a vehicle that is not where it is supposed to be, using cameras or sensors at the intersection. However, this method of detection is likely to be much slower to react to a problem. Each has advantages and disadvantages, and a combination of the two would most likely be the safest. The specifics of the implementation are beyond the scope of this analysis. What is important is that whenever a vehicle violates its reservation in any way, the intersection manager should become aware as soon as possible. Because our simulations only deal with collisions, we assume that the colliding vehicle sends a signal and the intersection manager becomes aware of the situation immediately.

As described in Chapter 3, our protocol includes a DONE message that vehicles transmit when they complete their reservations. One way to reliably sense when a vehicle is in distress would be to notice a missing DONE message. This approach has two drawbacks. First, the DONE message is optional, mainly because there is no incentive for the driver agent to transmit it. Second, the intersection manager may not be able to notice the missing message until some time after the incident has occurred. I describe the effects of this latency later in this chapter.

Informing Other Vehicles

We also assume that there exists a way for the intersection manager to broadcast the fact that something is wrong to the vehicles. Since the intersection manager can already communicate with the vehicles, this is not a big assumption, and in fact the EMERGENCY-STOP message as described in Chapter 3 does exactly this. For safety purposes, the mode of communication is slightly different from that employed in the rest of the communication protocol. Under normal operating conditions,

individual messages each containing multiple pieces of information are transmitted between agents. Because we cannot verify the receipt of these messages without a response, the semantics of the protocol ensure that whenever a message is sent, the sending agent makes the most conservative assumption—in the case of a REQUEST message, that it was not received; in the case of a CONFIRM message, that it was. In the event of a collision, however, the intersection manager needs to communicate one bit of information to as many vehicles as possible: that something is wrong. Because it is very important that all vehicles receive this message, it is transmitted repeatedly, to all vehicles, to make it as likely as possible that each vehicle receives the message. While we would like to assume that all vehicles receive this message, we will show in Section 9.3 that even when a significant number of vehicles do not, the safety measures in place still protect many vehicles that would otherwise wind up crashing.

9.2.2 Incident Mitigation

When a vehicle deviates significantly from its planned course through the intersection resulting in physical harm to the vehicle or its presumed occupants, we refer to the situation as an *incident*. Once an incident has occurred, the first priority is to ensure the safety of all persons and vehicles nearby. Because we expect such incidents to be very infrequent occurrences, re-establishing normal operation of the intersection is a lower priority and the optimization of that process is left to future work.

Intersection Manager Response

As soon as the intersection manager detects or is notified of an incident, it immediately stops granting reservations. All subsequent received requests are rejected without consideration. Due to the nature of the protocol, the intersection manager cannot revoke reservations, as driver agents would have no incentive to acknowledge

their receipt. However, the intersection manager can send a message to the vehicles that an incident has occurred. This message is the special EMERGENCY-STOP message, which the intersection manager may only send in an emergency situation, and which (as with the rest of the protocol) it must assume has not been received.

The EMERGENCY-STOP message lets vehicles know that an event has taken place in the intersection such that:

- no further reservations will be accepted
- vehicles able to come to a stop before entering the intersection should do so
- vehicles in the intersection should no longer assume that "near misses" will not result in collisions

For human-compatible policies, such as FCFS-SIGNAL, the intersection manager also turns all signals red. In a real-world implementation, a more conspicuous visual cue could be provided, but semantically it is only important that the intersection informs the human drivers that they may not enter.

Vehicle Response

For the EMERGENCY-STOP message to be useful in any way, driver agents must react to it. Here we explain the specific actions our implementation of the driver agent takes when it receives this message. Normally, when approaching the intersection, our driver agent ignores any vehicles sensed in the intersection. This is because what might otherwise appear to be an imminent collision on the open road is almost certainly a precisely coordinated "near-miss" in the intersection. However, once the driver agent receives the EMERGENCY-STOP message from the intersection manager, it disables this behavior. If the vehicle is in the intersection, the driver agent will not blindly drive into another vehicle if it can help it. If the vehicle is not in the intersection and can stop in time, it will not enter, even if it has a reservation.

While our first inclination was to make the driver agent immediately decelerate to a stop, we quickly realized that this is not the safest behavior. If all vehicles that receive the message come to a stop, vehicles that would otherwise have cleared the intersection without colliding may find themselves stuck in the intersection—another object for other vehicles to run into. Such an overreaction might be particularly bad if the vehicle that caused the incident is on the edge of the intersection where it is unlikely to be hit. Trying to stop all the other vehicles in the intersection just makes this situation worse.

If a driver agent does detect an impending collision, it should take evasive actions or apply the brakes. Since our protocol governs a true multiagent system with self-interested agents, we cannot prevent driver agents from doing so, even if it is detrimental to vehicles overall. Thus, our driver agent brakes if it believes a collision is imminent.

9.3 Experiments

In order to evaluate the effects of our reactive safety measures, we performed several experiments in which various components were intentionally disabled. The various configurations can be separated into three classes. An *oblivious* intersection manager takes no action at all upon detecting an incident. An intersection manager utilizing *passive* safety measures stops accepting reservations, but does not send any EMERGENCY-STOP messages to nearby driver agents. Finally, the *active* configuration of the intersection manager—which corresponds to the full version of the protocol as specified in Chapter 3—has all safety features in place. In addition to considering these three incarnations of the intersection manager, we also study the effects of unreliable communication in the active case. Note that when no vehicles receive the EMERGENCY-STOP message, the active and passive configurations are identical.

9.3.1 Experimental Setup

With the great efficiency of the reservation-based system comes an extreme sensitivity to error. While buffering might protect against minute discrepancies, it cannot hope to cover gross mechanical malfunctions. To determine just how much of an effect such a malfunction would have, we created a simulation in which individual vehicles could be "crashed" (given the CRASH disability from Chapter 6.3.4), causing them to immediately stop and remain stopped. Whenever a vehicle that is not crashed comes into contact with one that is, it becomes crashed as well. While this does not model the specifics of individual impacts, it does allow us to estimate how a malfunction might lead to collisions.

In order to ensure that we included malfunctions in all different parts of the intersection, we triggered each incident by choosing a random (x, y) coordinate pair inside the intersection, and crashing the first vehicle to cross either the x or y coordinate. This is akin to creating two infinitesimally thin walls, one horizontal and the other vertical, that intersect at (x, y). Figure 9.1 provides a visual depiction of this process.

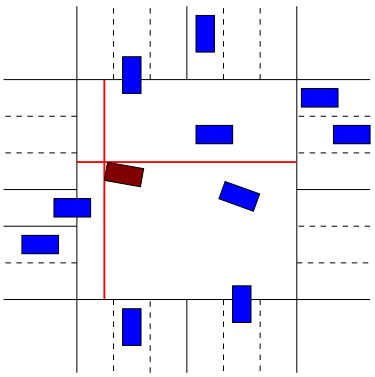


Figure 9.1: Triggering an incident in the intersection simulator. The dark vehicle turning left is crashed because it has crossed the randomly chosen x coordinate. If a different vehicle had crossed that x coordinate or the randomly chosen y coordinate earlier, it would be crashed instead.

After initiating an incident, we ran the simulator for an additional 60 seconds, observing any subsequent collisions and recording when they occurred. Using this information, we constructed a *crash log*, which is essentially a histogram of crashed vehicles. For each step of the remaining simulation, the crash log indicates how many vehicles were crashed by that step. By averaging over many such crash logs for each configuration, we were able to construct an "average" crash log, which gives a picture of what a typical incident would produce.

Because our system is compatible with humans, we included experiments with a human-compatible intersection control policy. As demonstrated in Chapter 8.3, when a significant number of human drivers are present, the FCFS-Signal cannot offer much of a performance benefit over traditional traffic signal systems. As such, we limited our experimentation to scenarios in which 5% of the vehicles are controlled by simulated human drivers, and used a Single-Lane signal model (see Chapter 8.1.1). With only 5% human drivers, an FCFS-Signal policy can still create a lot of the precarious situations that are the focus of this investigation.

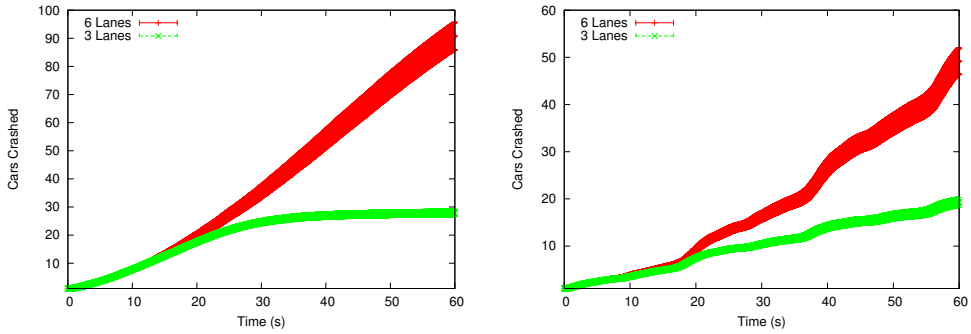
For these experiments, we ran our simulator with scenarios of 3, 4, 5, and 6 lanes in each of the four cardinal directions, although we will discuss results only for the 3- and 6-lane cases (other results were similar) for the sake of brevity. As with earlier experiments, vehicles are spawned equally likely in all directions, and are generated via a Poisson process which is controlled by the probability that a vehicle will be generated at each step. Vehicles are generated with a set destination—15% of vehicles turn left, 15% turn right, and the remaining 70% go straight. As before, the leftmost lane is always a left turn lane, while the right lane is always a right turn lane. Turning vehicles are always spawned in the correct lane, and non-turning vehicles are not spawned in the turn lanes. In scenarios involving only autonomous vehicles, we set the traffic level at an average of 1.667 vehicles per second per lane in each direction. This equates to 5 total vehicles per second for 3 lanes, and 10 total

vehicles per second for 6 lanes. Scenarios with human-driven vehicles had one third the traffic of the fully autonomous scenarios—the intersection cannot be nearly as efficient with human drivers present. We chose these amounts of traffic as they are toward the high end of the spectrum of manageable traffic for the respective variants of the intersection manager. While we wanted traffic to be flowing smoothly, we also wanted the intersection to be full of vehicles to test situations that likely lead to the most destructive possible collisions.

9.3.2 How Bad Is It?

As we suspected, the average crash log of the oblivious intersection manager is quite grisly. As explained in Section 9.2.2, driver agents must ignore their sensors while in the intersection, because many of the "close calls" would appear to be impending collisions. Without any way to react the situation going awry, vehicles careen into the intersection, piling up until the entire intersection is filled and crashed vehicles protrude into the incoming lanes. Figure 9.2 shows that for both 6-lane cases—fully autonomous and 5% human drivers—the rate of collisions does not abate until over 70 vehicles have crashed. Even a full 60 seconds after the incident begins, vehicles are still colliding. In the 3-lane case, the intersection is much smaller and thus fills much more rapidly; by 50 seconds, the number of collided vehicles levels off.

In both of the scenarios with human drivers, shown in Figure 9.2(b), the number of vehicles involved in the average incident is noticeably smaller. This outcome is likely the result of two factors. First and foremost, the FCFS-Signal policy must make broad allowances to accommodate the human drivers, and thus overall is inherently less dangerous. The characteristic "close calls" from the standard FCFS policy are less common. Second, the simulated human driver agents do not drive "blindly" into the intersection—trusting to the intersection manager—the way the autonomous vehicles do. Also of note in Figure 9.2(b) is the visible periodicity of the signal model portion of the policy. As paths open up for autonomous vehicles due



(a) All autonomous (b) 5% humans

Figure 9.2: Average crash logs (with 95% confidence interval) for 3- and 6-lane oblivious intersections. In 9.2(a), the intersection manages only autonomous vehicles, while 9.2(b) includes 5% human drivers.

to changes in the signals, they drive unwittingly into the growing mass of crashed cars.

9.3.3 Reducing the Number of Collisions

There are two main components to the safety mechanism introduced in this chapter. First, the intersection manager stops accepting reservations. Second, the intersection manager sends messages informing the driver agents that an incident has taken place. There is a possibility that this second part might not always work perfectly; some vehicles might not receive the message. To investigate the effects of these potential communication failures, we intentionally disabled some of the vehicles' ability to receive the EMERGENCY-STOP message. A parameter in our simulator controls the fraction of vehicles created with this property, and by varying this parameter, we could observe its subsequent effect on the average number of vehicles involved in incidents.

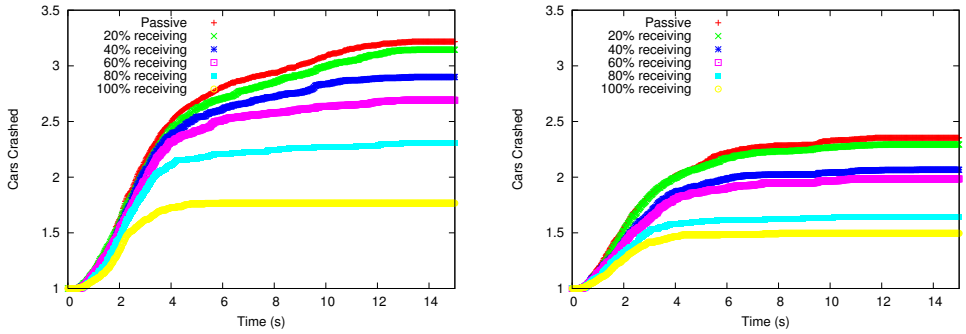
As compared to the oblivious intersection manager, the number of vehicles involved in the average incident for an active intersection manager decreases dramatically. Table 9.1 shows the numerical results for both the 3- and 6-lane intersections,

along with a 95% confidence interval. The average crash logs for these runs are not shown in Figure 9.2, as they would be indistinguishable from one another at that scale. Instead, we present them in Figure 9.3.

	Fully Autonomous		5% Human	
	3 Lanes	6 Lanes	3 Lanes	6 Lanes
Oblivious	**27.9** $\pm$ 1.3	**90.9** $\pm$ 4.9	**19.3** $\pm$ 1.1	**49.3** $\pm$ 2.7
Passive	**2.63** $\pm$ .13	**3.23** $\pm$ .16	**2.23** $\pm$ .10	**2.35** $\pm$ .13
Active				
20% receiving	2.44 $\pm$ .13	3.15 $\pm$ .17	2.07 $\pm$ .10	2.29 $\pm$ .13
40% receiving	2.28 $\pm$ .12	2.90 $\pm$ .16	1.91 $\pm$ .10	2.07 $\pm$ .12
60% receiving	1.89 $\pm$ .10	2.69 $\pm$ .15	1.72 $\pm$ .09	1.98 $\pm$ .11
80% receiving	1.71 $\pm$ .08	2.30 $\pm$ .13	1.46 $\pm$ .07	1.65 $\pm$ .09
100% receiving	**1.36** $\pm$ .06	**1.77** $\pm$ .10	**1.22** $\pm$ .05	**1.50** $\pm$ .09

Table 9.1: Average number of simulated vehicles involved in incidents for 3- and 6-lane intersections. Even with only the passive safety measures, the number of crashed vehicles is dramatically decreased from the oblivious intersection manager. In the active configuration, as more vehicles receive the emergency signal, the number of crashed vehicles decreases further.

Figure 9.3 shows the effects of the reactive safety measures in intersections with 6 lanes, with the proportion of receiving vehicles varying from 0% (passive) to 100% in increments of 20%. Even in the passive configuration, the overall number of vehicles involved in the average incident decreases by a factor of almost 30 in the fully autonomous scenario, and a factor of over 20 in the scenario with 5% human drivers, as compared to the oblivious intersection manager. As expected in the active configuration, when more vehicles receive the emergency signal, fewer wind up crashing. The graphs in Figure 9.3 only show the first 15 seconds of the incident, because in no case did a collision occur more than 15 seconds after the incident started.



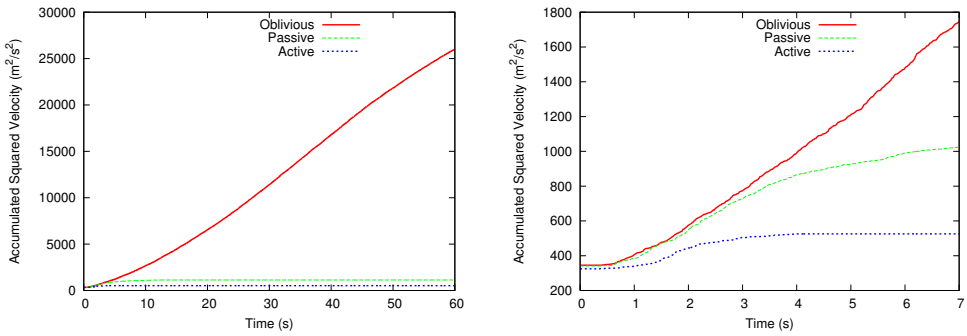
(a) All autonomous

(b) 5% humans

Figure 9.3: The first 15 seconds of average crash logs for 6-lane passive and active intersections. As more vehicles react to the signal, safety improves.

9.3.4 Reducing the Severity of Collisions

While it is reassuring to know that the number of vehicles involved in the average incident can be kept fairly low, these data do not give the entire picture. For example, compare an incident in which 30 vehicles each lose a hubcap to one in which two vehicles are completely destroyed and all occupants killed. While we do not currently have any plans to model the intricate physics of each individual collision with high fidelity, our simulations do allow us to observe the velocity at which the collisions occur. In the previous example, we might notice that the 30 vehicles all bumped into one another at low velocities, while the two vehicles were traveling at full speed. To quantify this information, we record not only when a collision happens, but the velocity at which it happens. In a collision, the amount of damage done is approximately proportional to the amount of kinetic energy that is lost. Because kinetic energy is proportional to the square of velocity, we can use a running total of the squares of these crash velocities to create a rough estimate of the amount of damage caused by the incident. Figure 9.4 shows an average "damage log" of a 6-lane intersection of autonomous vehicles. Qualitatively similar results were found for the other intersection types.



(a) the average incident

(b) zoomed in

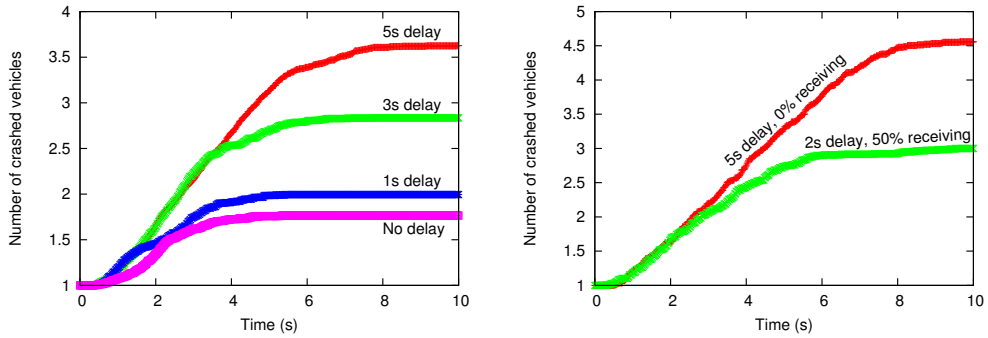
Figure 9.4: Average total squared velocity of crashed vehicles for a 6-lane intersection with only autonomous vehicles. Sending the emergency message to vehicles not only causes fewer collisions, but also makes the collisions that do happen less dangerous.

As Figure 9.4(a) shows, the effect of our safety measures under this metric is quite dramatic as well. In the passive case the total accumulated squared velocity decreases by a factor of over 25. In the active case, with all vehicles receiving the signal, it decreases by another factor of 2. Of particular note is the zoomed-in graph in Figure 9.4(b). In the passive configuration, the total squared velocity accumulates as if the intersection manager were oblivious, until the first vehicles stop short of the intersection at around 3 seconds; without a reservation, they may not enter. In the active scenario, when all the vehicles receive the message, the improvement is almost immediate.

9.3.5 Delayed Incident Detection

Implicit in these results is the assumption that intersection managers become aware of incidents instantaneously. While this could be the case in many collisions—vehicles should communicate when they have collided—if a vehicle's communications are faulty, or if the vehicle does not realize it has collided, the intersection may not discover the problem for a few seconds, when another vehicle or sensor will detect the problem. To assess the effects of delayed incident detection, we artificially delayed

the intersection manager's response in some of our simulations. Figure 9.5 shows the results from these experiments.



(a) delaying detection (b) delays and faulty communication

Figure 9.5: Crash logs showing the effects of delayed incident detection.

In Figure 9.5(a), the intersection manager's reaction was delayed 0, 1, 3, and 5 seconds. Note that the total number of crashed vehicles with a delay of 5 seconds is on par with the number in the experiment in which the intersection manager reacts immediately, but none of the vehicles receive the message, shown in Figure 9.3(a). Figure 9.5(b) shows what happens with both delayed detection and faulty communication. This graph, along with the earlier results, suggests that for small values, each second of delay is approximately equivalent to 20% of vehicles not receiving the EMERGENCY-STOP message, and that when combined, delayed detection and faulty communication have an additive effect. For larger delays, the number of vehicles involved can be approximated using the data shown in Figure 9.2(a), because in these cases, the number of vehicles that crash after the intersection is much smaller than the number that crash before it reacts.

9.4 Safety Discussion

The results in this section suggest that it may be possible to improve efficiency while also improving safety. But of course before deployment in the real world,

extensive testing with real vehicles would be needed in order to verify both the suggested efficiency benefits, as well as the safety properties of the system. People are often hesitant to put their well-being (physical or otherwise) in the hands of a computer unless they can be convinced that they will receive a significant safety benefit in exchange for surrendering precious control. Humans often suffer from the *overconfidence effect*, erroneously believing they are more skillful than others. In a 1981 survey of Swedish drivers, respondents were asked to rate their driving ability in relation to others. A full 80% of those asked placed themselves in the top 30% of drivers [Svenson, 1981]. It is this effect that creates the high standard to which computerized systems are held. It is insufficient for such systems to be marginally safer, or safer for the *average* user; they must be the very paragon of safety.

In our experiments, we showed that the number of vehicles involved in individual incidents can be drastically reduced by utilizing a fairly straightforward reactive safety mechanism. In fact, in the active configuration with 3 lanes, 75% of the incidents involved only one vehicle: the one we intentionally crashed (60% for 6 lanes). If this was a vehicle with a mechanical failure, no other vehicles would have crashed into it! Even in the passive case with 6 lanes of traffic, an average of only 3.23 vehicles were involved. But how does this compare with current systems? If we make the overly conservative assumption that accidents in traffic today involve only one vehicle, this represents a 223% increase in vehicles-per-incident. However, autonomous vehicles should prevent a lot of accidents because they will all but eliminate driver error. So, even with an increased number of vehicles involved in each incident, if the total number of incidents can be reduced by just 70%, these experiments suggest that an autonomous intersection management system will be safer overall. A 2002 report for the U.S. Federal Highway Administration blamed over 95% of all accidents on driver error [Wierwille *et al.*, 2002]. The remaining accidents were divided equally between vehicle failures and problems with roads.

It is important to note that these numbers are for all driving, not just intersection driving. Accidents in intersections are even more likely to be caused by driver error, sometimes even by drivers willfully disobeying the law: running red lights and stop signs or making illegal "U"-turns.

Even if we make overly conservative assumptions—that all driving is as dangerous as intersection driving, and that driver error is no more accountable for intersection crashes than it is in other types of driving—our data suggest that automobile traffic with autonomous driver agents and an intersection control mechanism like ours will reduce collisions in intersections by over 80%. We believe that in reality, the improvement will be much greater.

The safety measures presented in this section constitute just one approach for mitigating the system's failure modes. More sophisticated methods involving explicit cooperation amongst vehicles may create an even safer system. We have not shown (or attempted to show) that this particular solution is the best possible. Rather we have demonstrated that even with a simple and straightforward response to accidents, the overall safety of the system can be maintained, without sacrificing the benefits of vastly improved efficiency.

Chapter 10

Multiple Intersections

In this section, we propose a novel augmentation that allows the reservation-based intersection control mechanism to work for more than just a single intersection.

10.1 Challenges

In a single-intersection scenario, once a vehicle has completely cleared the intersection, the intersection manager no longer bears any responsibility for the vehicle. However, when two or more intersections are linked together, the problem with this notion becomes more apparent. By granting a reservation, an intersection manager is guaranteeing that the vehicle will be safe it if follows the parameters of the reservation, which may include specific directives for acceleration throughout the traversal. When a vehicle has completed the traversal, it may be traveling at significant speed. If vehicles are stopped or slowed just outside the intersection, it may not be able to decelerate rapidly enough to prevent a collision.

10.2 The Admission Control Zone

To prevent such a situation, we introduce the concept of an *admission control zone* (ACZ). The ACZ, although it includes the word "admission" in its name, is actually positioned *after* the intersection. It is called an *admission* control zone because

admission to the zone is controlled. The ACZ acts like a "leaky bucket" (an *admission control protocol*) from computer communication networks [Turner, 1986]. Each ACZ has a fixed *capacity*, measured in meters, as well as a *distance*. The capacity must be no larger than the distance, and in practice, should be significantly smaller, perhaps half the size of the distance. When a vehicle obtains a reservation that departs the intersection in a particular lane, it also has a space reserved for it in that lane's ACZ. As part of the confirmation, the intersection manager reveals the ACZ distance (but not capacity) to the vehicle. When a vehicle travels beyond the ACZ distance, it sends a message back to the intersection manager that releases the space reserved for it in the ACZ. If the ACZ of a vehicle's requested departure lane does not have sufficient remaining capacity for the vehicle (including stopping distance), the vehicle's request will be rejected, even if the necessary space-time in the intersection was available. Figure 10.1 illustrates one potential scenario.

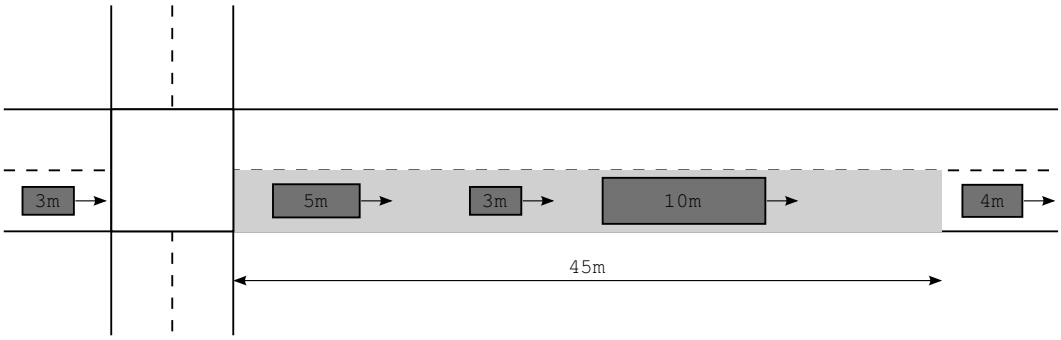


Figure 10.1: The light gray area depicts the ACZ for one lane. The 5m, 3m, and 10m vehicles are in the ACZ, while the 4m vehicle has left the ACZ. The ACZ *distance* is 45m. Let the ACZ *capacity* be 25m. The total length of vehicles in the ACZ is currently 18m. The 3m vehicle approaching the intersection cannot obtain a reservation departing in its current lane unless it can stop within 4m after entering, or the 10m vehicle departs the ACZ.

10.2.1 Lane Changing Within The ACZ

Because the intersection must monitor the total length of all vehicles in each ACZ, lane changing within the ACZ distance must be carefully controlled or forbidden altogether. If a vehicle changed into a lane that otherwise would have had enough room for a requesting vehicle, that vehicle's reservation may be confirmed, even though there may now not be enough room for the vehicle to exit the intersection safely. Forbidding lane changing altogether may not be feasible, as some vehicles may need to change lanes in order to exit the roadway, or to prepare for an upcoming turn. Instead, we control lane changing inside the ACZ using special messages: ACZRequest, ACZConfirm, ACZReject, ACZCancel, ACZDone, ACZEntered, and ACZExit. These messages—presented along with the rest of the complete protocol in Chapter 3, and marked with a † symbol—allow a vehicle within the ACZ distance to request permission from the intersection to change lanes or enter the roadway in a particular lane. They also enable a vehicle to inform the intersection that it has completed changing lanes or left the roadway before clearing the ACZ.

10.2.2 Data Structure

To enable the ACZ to function robustly and efficiently, we developed a custom data structure. This data structure has at its heart a heavily modified queue, which represents the vehicles currently located physically inside the ACZ. The queue is implemented as a doubly-linked list with a tail but no head; while nodes are put onto the end of the list, they are never removed directly from the front. Each node in the list represents a vehicle and contains the vehicle's VIN and length. The nodes in the queue come in two variants: those that were *enqueued*, and those that were *inserted*. If a node is enqueued, all nodes in front of it represent vehicles that are physically in front of the vehicle it represents. A node is enqueued if and only if the

vehicle it represents enters the ACZ from the intersection—not by a lane change.

In addition to the nodes in the queue, the data structure has *pending* nodes. Pending nodes represent vehicles with space reserved in the ACZ, but not physically located in the ACZ. This space may be reserved either as part of the reservation process or as part of a lane-change request inside the ACZ. When those vehicles enter the ACZ, the nodes are enqueued or inserted depending on how the vehicle entered the ACZ. The vehicle lengths in all nodes contribute to the total length of all vehicles in the ACZ, which may never exceed the ACZ's capacity. When a node is created, the total length is increased by the length in the new node. When a node is destroyed, the total length is correspondingly decreased.

Supporting the queue and the pending nodes are two maps. The first maps VINs to nodes, allowing constant-time access to any node. The second assists with enqueueing vehicles that will enter the ACZ in accordance with a reservation. When a reservation is requested, the intersection manager checks if the ACZ has enough space. If so, a pending node is created, and an entry is made in the second map, indicating the time at which the vehicle will enter the ACZ. Periodically—ideally very frequently, at a minimum just before any nodes are added to the queue—the second map is searched for all keys before the current time, which are then enqueued.

As with adding nodes, there are two ways to remove nodes. Nodes can be *expired* or *extracted*. A node is expired when the corresponding vehicle leaves the ACZ by reaching the ACZ's distance from the intersection, sending an AWAY message. When a node is expired, if that node was enqueued, it is removed from the queue along with every node in front of it—the corresponding vehicles are physically in front of the vehicle represented by the node we expired, and thus must also have left the ACZ. If the node to be expired was inserted, we cannot make this guarantee, and the node is simply removed from the queue. When a node is extracted, on the other hand, it is simply removed from the queue regardless of how it was added. Nodes

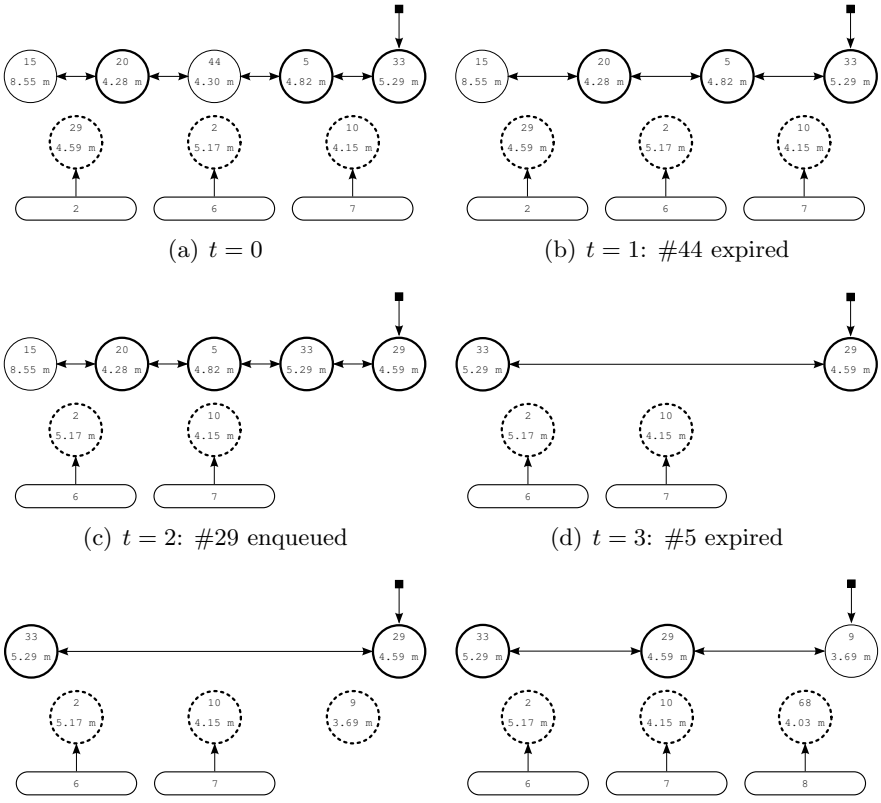
are extracted when the vehicle to which they correspond exits the ACZ by changing lanes or exiting the roadway, sending a ACZEntered or ACZExit, respectively. Pending nodes can also be removed, if the corresponding vehicles leave the ACZ in any way before those nodes have been added to the queue, or if the reservations or lane-change requests for which the nodes were created are canceled, via a Cancel or ACZCancel message, respectively.

Figure 10.2 illustrates the state of the ACZ data structure over the course of several operations—bold nodes are those that were added sequentially, dashed nodes are pending nodes, and the tail of the queue is indicated by the ■ symbol. Figure 10.2(a) shows the initial state, with five nodes in the queue, and three pending nodes, each of which will be enqueued sequentially at or after the times indicated. In 10.2(b), vehicle 44, which was inserted, is expired. By Figure 10.2(c), time 2 has passed, triggering the sequential enqueueing of the pending node for vehicle 29. In Figure 10.2(d), vehicle 5, which was enqueued, is expired, triggering the removal of its node and all nodes in front of it. Figure 10.2(e) shows that vehicle 9 has requested and been approved for a lane change into the ACZ, creating a new pending node. Finally, in Figure 10.2(f) vehicle 9 has completed its lane change and its node has been inserted. Its node was not enqueued, as it may be in front of one of the other vehicles. Additionally, vehicle 68 has requested and obtained a reservation and will enter the ACZ at time 8. For clarity, the maps that allow these operations to take place in amortized constant time are not shown in the figure.

With the addition of the ACZ, we have identified and solved the main technical barrier between a system capable of only single intersections, and one that works at networks of many intersections.

10.2.3 Light-Based and Human-Usable Policies

In signal-based policies, as well as any that accommodate vehicles without the capability to communicate, the utility of the ACZ is significantly diminished. In



(a) $t = 0$

(b) $t = 1$: #44 expired

(c) $t = 2$: #29 enqueued

(d) $t = 3$: #5 expired

(e) $t = 4$: #9 approved for lane change

(f) $t = 5$: #9 completes lane change, #68 obtains reservation

Figure 10.2: The internal state of the ACZ data structure over the course of several operations.

these cases, vehicles cannot be carefully tracked, and thus the ACZ cannot make the guarantees present in a fully-autonomous case. In these cases, it must be the responsibility of each individual driver agent to assess the outflow of the lane in which it plans to depart the intersection to ensure there is sufficient capacity. These otherwise non-coordinating vehicles might be augmented to allow rudimentary coordination, even with a human driver. If the vehicle can sense when it is entering an ACZ, when it is exiting an ACZ, and when the vehicle wants to change lanes within the ACZ (perhaps by examining the state of the turn signal), ACZ might retain its utility in mixed-population scenarios. However, all vehicles would need to have this baseline capability.

10.3 Experimental Results

To test our protocols, we used our custom traffic simulator, with traffic level $0 < \lambda \leq 0.2$ in the V2I scenarios, and $0 < \lambda \leq 0.08$ in the V2V scenarios, which are for lighter-traffic intersections. Recall that λ is the rate parameter of the Poisson process that generates the traffic, so there will be an average of λ vehicles per second in each lane. Once vehicles are spawned, they are assigned a destination. In some experiments, the destination is assigned randomly. In others, the destination is assigned so as to prevent vehicles from needing to turn. Each lane has a speed limit of 25m/s. A video of the `aim3` simulator running the FCFS policy at multiple intersections can be seen on the videos section of the project page at `http://www.cs.utexas.edu/~kdresner/aim/`.

10.3.1 Delay

As with previous experiments, the metric we consider is *delay*—the total increase in travel time due to the presence of the intersection. However, in this case, instead of simply reporting total average delay, we report delay *per intersection*—each vehicle's total delay is divided by the number of intersections traversed by that vehicle.

This normalization is to account for the fact that some vehicles may traverse fewer intersections than others.

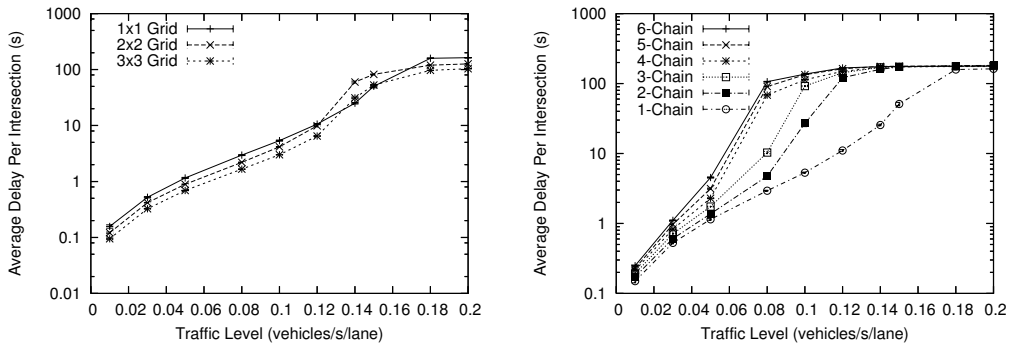
10.3.2 V2I Results

We considered two different topologies for multiple intersections: grids and chains. The results are shown in Figure 10.3. In 10.3(a), we see that for the most part, delay per intersection actually *decreases* as the size of the grid increases. The exception for $0.12 \leq \lambda \leq 0.18$ may be an artifact of the ACZ system. If an intersection gets significantly backed up (as may happen at higher traffic levels), the ACZ for an "upstream" intersection may get clogged, causing vehicles to wait even though the intersection would otherwise be clear. This situation does not arise when there is only one intersection. The plots level out toward the right end of the graph as the simulator simply has no more room to spawn vehicles.

But why should the delay per intersection decrease as the number of intersections increases? As traffic passes through the managed intersections, parts of the mechanism designed to keep vehicles from coming too close to one another have the effect of spreading out the traffic amongst the lanes. The intersection manager won't let a vehicle exit a lane too soon after another vehicle has done so. Once the traffic is spread out by one intersection, it is less likely to need to do so for later intersections, which results in much lower delays at the later intersection. As the grid size increases, more intersections have more roads entering them with such traffic.

Figure 10.3(b) shows results from chains of size 1–6, in which we see the opposite pattern. Delay per intersection *increases* as the chain grows. The effect mentioned above is much weaker in a chain, as adding another intersection to the chain adds more "noisy" traffic and very little "shaped" traffic. Additionally, the chain topology has the pronounced effect of concentrating more and more traffic on the intersections toward the middle of the chain due to the random assignment of

destinations. In one experiment a single intersection was crossed 21,600 times. With the same settings and duration, each middle intersection in a 6-chain had almost 50,000 crossings. This effect is absent in grids because the ratio of lanes (which spawn traffic) to intersections is constant.



(a) Grids with turning

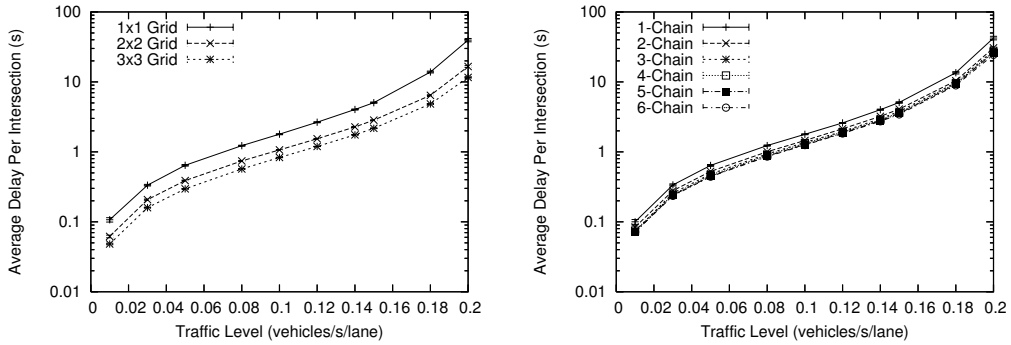
(b) Chains with turning

Figure 10.3: Average delay per intersection for V2I grids and chains. The y-axis is a log scale.

To test these hypotheses, we ran experiments in which vehicles did not turn. The results are shown in Figure 10.4. As 10.4(a) and 10.4(b) show, delays are much lower when turning is disabled. Turning vehicles must slow down to make the turns, use significantly more space-time in the intersection, and can interfere with many more vehicles' trajectories. But the absolute numbers are not the interesting part—the fact that the delays decrease as the size of the chain grows helps confirm our hypotheses. Without turning, traffic is no longer concentrated on the middle intersections, and the "shaping" effect on the long lane decreases delay.

10.3.3 V2V Results

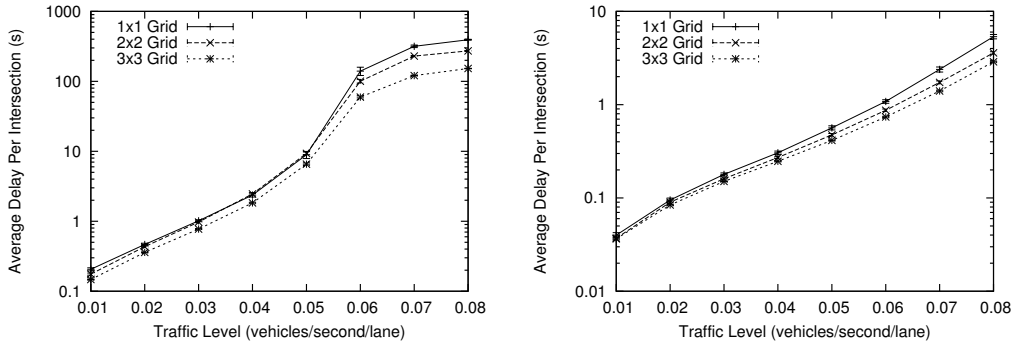
Figure 10.5 shows results from our V2V experiments on grid topologies. Just as with the V2I system, increasing the number of intersections decreases the delay per intersection. The V2V protocol has a similar tendency to spread vehicles out and to perform better when vehicles are already spread out. Recall vehicles with



(a) Grids without turning (b) Chains without turning

Figure 10.4: Average delay per intersection for V2I grids and chains without turning enabled. The y-axis is a log scale.

intersecting trajectories—such as those exiting in the same lane—cannot be in the intersection at the same time. Also note that in 10.5(a), the anomaly from the V2I case is absent, because the V2V system does not have an ACZ. We also explored the chain topology for the V2V system. The results were qualitatively similar to those from the V2I system: enlarging the chain increased delay per intersection, but the effect vanished with turning disabled.



(a) Grids with turning (b) Grids without turning

Figure 10.5: Average delay per intersection for V2V grids. The y-axis is a log scale

10.3.4 Mixing V2I and V2V

While all the experimental results presented in this section are for homogeneous systems of intersection managers, there is no requirement that such homogeneity exist. As explained in Chapter 5.2.1, when a vehicle approaches an intersection, it first determines what type the intersection is (V2I or V2V), so that it knows which protocol to use. This means that any configuration of intersections can work, although the V2V intersections cannot receive as much traffic as a comparable V2I intersection can.

10.4 Summary

This chapter proposed a sophisticated extension to the basic intersection control mechanism capable of supporting more than a single intersection. In it, I described some of the important challenges that result from having multiple intersections. By introducing the ACZ, I enabled the intersection manager to grant reservations while still ensuring that vehicles would not be placed in danger by traversing the intersection according to those reservations. Furthermore, this chapter provided experimental results showing that the mechanism is effective. While there is still ample space to explore regarding the effects of multiple intersections, this chapter takes one more step toward making autonomous intersection management a reality.

Chapter 11

Related Work

Traffic control is a vast area of research for computer scientists and engineers alike. The field of Intelligent Transportation Systems (ITS) is concerned with applying information, computing, and sensor technologies to solve problems in traffic and road management [Bishop, 2005]. ITS includes intelligent vehicles (IV) as well as infrastructure, such as intersections. Unfortunately, while both aspects of ITS are heavily studied, relatively little current research considers how intelligent or autonomous vehicles and infrastructure can work together to improve the efficiency and safety of the overall traffic system. The Berkeley PATH project has produced a lot of interesting work, including work on a fully-automated highway [Alvarez and Horowitz, 1997].

In this section, we describe some work related to our own, both directly and tangentially. Some of this work is specifically concerned with intersection control, some takes a multiagent approach to other aspects of traffic management, and some represents work on the technologies necessary to bring fully autonomous vehicles into the mainstream.

11.1 Requisite Technology

Before autonomous vehicles can take over the roads, they will need to be able to interact with all the aspects of roadways, including pedestrians, other vehicles, and lanes. As early as 1991, a driver agent system named "Ulysses" had been developed in simulation [Reece and Shafer, 1991]. While most systems currently under development for implementation on real vehicles are geared toward assisting human drivers, many of the technologies created through these efforts are applicable to the creation of a completely autonomous driver agent. Such a successful driver agent needs to do three main things: detect other entities on the road, keep its vehicle in the lane, and maintain safe distances from other vehicles. Fortunately, each of these three subtasks currently attracts an extensive amount of research.

11.1.1 Object Detection and Tracking

A fully autonomous vehicle must be able to reliably detect, classify, and track various objects that may be in the roadway. From pedestrians and bicycles to cars and trucks, autonomous vehicles will require robust sensors that can monitor the world around them in all manner of lighting conditions and weather. Without such abilities, any amount of higher reasoning a driver agent can do is irrelevant. Fortunately, researchers are attacking this problem with many techniques.

In 2004, Honda introduced an intelligent night vision system to the Japanese market capable of detecting pedestrians [Liu and Fujimura, 2003]. The system uses two far-IR (FIR) cameras on the front of the vehicle to detect heat-emitting objects beyond the range illuminated by the vehicle's headlights. The two cameras allow the system to obtain distance information about the detected pedestrians and can then warn the driver. DaimlerChrysler is developing a similar system that also extrapolates the trajectories of classified objects in order to predict possible outcomes sooner [Gavrila *et al.*, 2004]. Mählisch et al. [2005] have developed a

sensor fusion technique that can glean information about pedestrians reliably even from low-resolution images.

The Ford Motor Company has been investigating how to track vehicles using both color and shape information [She *et al.*, 2004]. Gepperth et al. [2005] have demonstrated that with only gray-valued videos (no color), a two-stage (initial detection and confirmation) mechanism using a simple neural network for confirmation can reliably and quickly classify other vehicles.

Vehicle and pedestrian classification and tracking is a well-studied area of IV research that is progressing quickly. A glance at any IV-related conference or symposium will reveal a plethora of articles aimed at using lidar, FIR, normal video, and any combination of these sensors with algorithms like Kalman filters, particle filters, and neural networks to track and classify other objects on the road.

11.1.2 Lane Following

As with pedestrian and vehicle detection and tracking, lane following is a heavily studied area of IV research. Varying from passive lane- and road-departure warning systems (LDWS/RDWS) to active lane keeping assistance (LKA), many systems are already showing up in production vehicles.

As far as RDWS go, Kohl et al. [2006] have used neuroevolution to create a warning system that can warn drivers of both road departure and impending crashes with other vehicles. The system was tested both in simulation and with a robotic vehicle. This work is sponsored by Toyota, who have also currently have an LDWS on the market in Japan. This system is unique in that it uses a rear-facing camera to predict and warn of impending lane departures. While LDWS and RDWS promise extensive benefits to drivers, they only warn of imminent road and lane departures, and do not provide information on what specific action should be taken. Autonomous vehicles will need to ensure they do not reach a point where a lane or road departure is imminent.

Lane keeping, on the other hand, provides and executes actions. For example, the "No Hands Across America" project in 1995 drove a vehicle 2,849 miles from Pittsburgh to Los Angeles. For 98.2% of the journey, the vehicle steered itself [Pomerleau, 1993]. More recent projects have concentrated on making such systems robust to varying speed, inclement weather and poor lighting conditions such as beneath overpasses and in tunnels. Wu et al. [2005] have proposed and tested a vision-based lane-keeping system that can operate at varying speed while providing smooth human-like steering. Watanabe and Nishida [2005], working for Toyota, have developed a lane detection algorithm specifically designed for steering assistance systems that is extremely robust to varying road conditions and lighting.

While several LKA systems are on the market in Japan, these systems are not intended to allow autonomous driving. Rather, they attempt to reduce driver fatigue and make turning more stable [Bishop, 2005]. Production systems that allow autonomous steering are almost invariably based on specially painted lines and are limited to special vehicles on closed courses.

Even without the benefit of explicitly designated lanes, autonomous vehicles can keep themselves on the roadway. In the 2005 DARPA Grand Challenge [DARPA, 2007a], the winning vehicle, "Stanley", used a technique fusing short-range laser range finders with long-range video cameras to follow a rough dirt path. First, the vehicle found smooth areas in front of it using the laser range finders. Then it mapped this information onto video images from forward-facing cameras. By determining the color of the area in the image corresponding to the smooth areas found by the laser range finder, Stanley was able to extrapolate using a flood-fill-type algorithm to find which areas of the video image were on the dirt path [NOVA, 2006]. Ramström and Christensen [2005] achieved a similar goal by using a strategy based on a probabilistic generative model.

11.1.3 Adaptive Cruise Control

If lane-keeping systems represent the main lateral component of an autonomous vehicle's driver agent, then adaptive cruise control (ACC) is the main longitudinal component. ACC allows a vehicle to maintain a safe following distance and can react quicker than a human driver in the case of sudden deceleration by the vehicle in front. ACC systems are already available on the market—DaimlerChrysler's Mercedes-Benz S-class, for example, comes with a system that will automatically apply the brake if it detects that the driver is not slowing sufficiently fast. Jaguar, Honda, and BMW offer similar systems. Nissan and Toyota have recently begun offering "low-speed following" systems, which can follow other vehicles in slower, denser, urban traffic scenarios [Bishop, 2005]. ACC relies on robust sensing and uses radar, lidar, and traditional machine vision algorithms. By combining various "flavors" of ACC — low speed, high speed, etc.—an agent could control the longitudinal motion of a vehicle in all situations. Recently, the notion of *cooperative* adaptive cruise control (CACC) has emerged [Laumônier *et al.*, 2006]. This concept goes much further toward realizing the goal of fully autonomous vehicles. By allowing vehicles to collaborate and take advantage of the precision of autonomous driver agents, vehicles can use the existing road space much more efficiently.

11.2 Intersection Collision Avoidance

To date, much of the ITS work relating to intersections has focused on Intersection Collision Avoidance (ICA). This work seeks to warn the driver when the vehicle may be entering an intersection unsafely. With the aid of high-precision digital maps and GPS equipment, the vehicle detects and classifies the state of the traditional signaling systems placed at the intersection [Lindner *et al.*, 2004]. ICA systems typically do not take any action on behalf of the driver, but simply provide a visual or auditory warning.

Rasche and Naumann [1997; 1997; 1998] have worked extensively on decentralized solutions to intersection collision avoidance problems, including those involving autonomous vehicles. This work is very similar to ours in that it uses "potential points of collision" to restrict access to the intersection. Only one vehicle may occupy any potential point of collision at a time. Vehicles attempt to obtain a token (similar to a token-ring in computer networking) for each point needed to cross the intersection. Once a vehicle has all the necessary tokens, it may cross. Rasche and Naumann's system also includes a priority model that allows emergency vehicles to cross more quickly and prevents deadlocks amongst normal vehicles. However, the system fails to satisfy several of our desiderata. It does not make any guarantees, nor do the authors provide any results regarding the efficiency of the system as compared to a traditional system. Furthermore, the distributed algorithm is not shown to be resilient to unreliable communication. The authors also do not provide any insight into how the system could be adapted to work with a mixed human/autonomous vehicle population. The most striking difference, however, is that the mechanism does not seem to have any notion of planning ahead. Tokens for the potential points of collision are either taken or not taken—a vehicle can not seek to obtain a token for some point in the future, thus allowing it to proceed toward the intersection without slowing down while other vehicles have the tokens.

In the context of video games and animation, Reynolds [1999] has developed autonomous steering algorithms that attempt to avoid collisions in intersections that do not have any signaling mechanisms. Such a system would have the enormous advantage of not requiring any special infrastructure or agent at the intersection—vehicles equipped with such algorithms could operate at any intersection. Unfortunately, the two main drawbacks of the system make it unsuitable for use with real-life traffic. First, the algorithm does not let the agent choose which path it will take out of the intersection; a vehicle may even find itself exiting the intersection the

same way it came in, due to efforts to avoid colliding with other vehicles. Second, the algorithm only *attempts* to avoid collisions—it does not make any guarantees about safety.

Cooperative intersection collision avoidance is a form of *cooperative vehicle-highway system* (CVHS) in which the intersection is allowed to participate in the ICA problem. ICA systems contained entirely in individual vehicles cannot account for gaps in sensor views or other sources of incomplete information. Thus, a CVHS approach is required. As with many other ITS technologies, production systems still assume a human driver and attempt to warn them when a violation is about to occur, or in some cases, punish them after the fact, as with cameras that detect when a vehicle has run a red signal and automatically issues the driver a citation. The U.S. Department of Transportation is sponsoring several ICA projects including both infrastructure-only and cooperative approaches [USDOT, 2003]. The intention is to first deploy the infrastructure-only systems, and then as the market penetration of ICA-equipped vehicles increases, to roll out the cooperative systems. Significant work on ICA is also underway in Japan [Bishop, 2005].

While these systems are a large step toward enabling autonomous vehicles to take to the roads, none are designed to work specifically with autonomous vehicles. With the exception of the algorithm designed for games, each assumes both a human driver and traditional signaling systems—a clumsy, inefficient interface that will find itself all but obsolete due to autonomous vehicle technology.

11.3 Optimizing Traffic Signal Timing

The vast majority of deployed technology for intersection control involves calibrating the timing of traditional traffic signals in order to create a "wave of green" such that once vehicles reach one green signal, they continue through all subsequent intersections without having to stop. Unfortunately, in practice, such waves tend to

be sporadic and short-lived due to rapidly changing traffic patterns. However, they do offer substantial benefits compared to systems without this coordination.

TRANSYT, the Traffic Network Study Tool, is an off-line system that, given average traffic flows, can determine optimum fixed-time coordinated traffic signal timings [Robertson, 1969]. TRANSYT requires extensive data gathering and analysis, but is used very heavily all over the world. Unfortunately, this system is very brittle because it does not have the ability to react to unusual changes in traffic flow. For example, at the end of a major sporting event, thousands of vehicles may all be attempting to cross an intersection in a direction which under normal circumstances is rarely used. Because the signal timings are set up to reflect these normal circumstances, the length of time for which the departing vehicles get a green signal may be significantly less than the cross traffic, of which there may be little.

SCOOT, the Split, Cycle, and Offset Optimisation Technique, represents an advancement over TRANSYT [Hunt *et al.*, 1981]. SCOOT is an on-line adaptive traffic control system that can react to changes in traffic levels, give priority to vehicles such as buses, and even estimate vehicle emissions. While SCOOT has been shown to reduce traffic delays by an average of 20% over systems like TRANSYT, it still relies on traditional signaling systems and vehicles. Furthermore, SCOOT requires reliable traffic data in order to adapt, and thus may be slow to react to changes in traffic flow.

The more recent RHODES system, developed at the University of Arizona, actually predicts future traffic conditions based on detectors such as induction loops and video cameras, and outputs optimized signal timings for the predicted traffic conditions [Mirchandani and Wang, 2005]. RHODES takes advantage of modern communication and processing infrastructure to act quickly on new data about changing traffic conditions.

11.4 MAS and Traffic

Automobile traffic is a great example of a multiagent system, and it is not surprising that there is a lot of research into modelling and studying traffic using multiagent techniques. Many of these approaches consider systems consisting only of traffic-signal-controlling agents or driver agents, as opposed to a heterogeneous multiagent system with many kinds of agents. Nevertheless, many of the ideas involved could potentially be adapted to work within the framework of the reservation system.

11.4.1 Cooperative Traffic Signals

Much of MAS traffic research focuses on improving current technology (systems of traffic signals). For example, Roozemond [1999] allows intersections to act autonomously while sharing the data they gather. The intersections then use this information to make both short- and long-term predictions about the traffic and adjust accordingly. This strategy attempts to overcome one of the weaknesses of SCOOT: the need for large amounts of reliable traffic data. If multiple intersections can share data, each intersection will get a more accurate picture of the current traffic situation.

Bazzan [2005] has used a decentralized approach combining MAS and evolutionary game theory. The approach models each intersection as an individually-motivated agent which must focus not only on local goals (getting vehicles through the intersection), but also on global goals (reducing travel times for all vehicles). Both Bazzan and Roozemond's techniques still assume traditional signaling mechanisms and human drivers.

11.4.2 Platoons

In addition to multi-intersection systems, multi-vehicle systems are the focus of a lot of research. Much of this research centers on creating *platoons* of vehicles in order to

minimize the effects of stop-and-go driving. Consider a line of cars stopped at a red signal. When the signal turns green, the first car begins to move. Eventually, the car behind it notices that it has enough space to accelerate as well. Some time later, the vehicle at the back of the line will begin to move, but this may be too late to actually get through the intersection during the current green phase of the signal. If, on the other hand, all the vehicles were to simultaneously and uniformly accelerate, more vehicles could make it through each green phase, because the vehicles would more efficiently use the space-time available to them to cross the intersection.

Clement [2002] has proposed a model called "Simple Platoon Advancement" (SPA), which addresses this exact problem. SPA boasts the ability to get nearly twice as many vehicles through a green signal (increasing the signal's throughput) as compared to normal human drivers, in addition to any safety and delay benefits associated with automated control. One the vehicles are through the intersection and dispersed to safe following distances, control is returned to the human driver.

Hallé and Chaib-draa [2005] have used the platoon approach to facilitate collaborative driving in general. They allow vehicles, which are controlled by separate agents, to form such platoons, with varying degrees of autonomy. Vehicles merge and split with platoons using carefully crafted maneuvers, during which each vehicle in the platoon has a specific responsibility. They present both centralized version, in which a *master* vehicle gives orders to the rest of the platoon, and a decentralized version, in which social laws dictate each agent's role, while the platoon's leader acts only as a representative to other platoons.

Both platooning systems assume automated control of vehicles, but use ordinary traffic signals for intersection control. By using platoons, these methods attempt to solve a problem inherent in the traffic signals themselves—they are designed for humans to use, and are not well suited to automated vehicle control. The work presented in this article attempts to free autonomous vehicles from the

control of traffic signals and instead design a new system that specifically utilizes the capabilities of fully autonomous vehicles.

11.4.3 History-Based Traffic Control

Taking a different approach to intersection control, Balan and Luke [2006] use a history-based method to maximize *fairness* (all vehicles experience similar delays) as opposed to efficiency (the average vehicle experiences short delays). Under this paradigm, vehicles which have historically (previously in their journey) experienced long delays should be more likely to experience shorter delays at subsequent intersections. In addition to being a multi-intersection approach, this method uses a marketplace model involving a system of *credits* that can be given and taken in exchange for shorter and longer delays, respectively. Coordination at individual intersections is still done with traditional traffic signals, the timings of which are part of the mechanism. Interestingly, the fairness approach actually yields results that are also reasonably efficient.

11.5 Machine Learning and Traffic

Abdulhai et al. [2003] have used Q-learning, a simple, yet powerful form of reinforcement learning, to do on-line adaptive signal control. In the work, the authors explore both an isolated intersection as well as a linear chain of intersections. They demonstrate that Q-learning can significantly reduce delays for vehicles and quickly adapt to changing traffic patterns. Bull et al. [2004] have shown how Learning Classifier Systems (LCS) can also make traditional traffic signals more efficient. Wiering [2000] has demonstrated that multiagent, model-based reinforcement learning can also be used to optimize signal timings in more complex networks of intersections.

Kuyer et al. allow individual signals to be controlled by distinct, but networked agents. The agents coordinate using the max-plus multiagent reinforcement learning algorithm, which computes optimal joint actions via messages sent between

adjacent agents in the network. With probability $1 - \epsilon$, this action is taken, while with probability ϵ, an exploratory random joint action is taken.

While not focusing on intersections, Moriarty and Langley [1998] have shown that reinforcement learning—specifically neuro-evolution—can train efficient driver agents for lane, speed, and route selection during freeway driving, all of which are critical components for a fully autonomous vehicle. Additionally, many of the object tracking and detection examples mentioned previously use neural networks to classify objects.

11.6 Physical Robots

On real autonomous vehicles, Kolodko and Vlacic [2003] have created a small-scale system for intersection control which is very similar to the granularity-1 FCFS policy. The authors developed the mechanism for small Cooperative Autonomous Mobile Robots (CAMRs), which are about 30 cm in diameter and have a top speed of 10 cm/s. The CAMRs were programmed to follow Australian traffic laws, and communicate with several different types of messages. Once demonstrated on the CAMRs, the mechanism was scaled up to use IMARA vehicles, which are much larger (capable of carrying two human passengers) and faster (top speed of 30 km/h). The system is completely distributed and does not require extensive infrastructure at the intersection. However, it does assume that all vehicles cooperate with one another.

The DARPA Urban Challenge, the next evolution of the Grand Challenge discussed previously, pits real autonomous vehicles against one another in an array of driving tasks in an urban setting [DARPA, 2007a]. Much like an autonomous vehicle "driver's test," it requires competing vehicles to navigate streets amidst other vehicles, parallel park, make a three-point (or more!) turn, all while following the appropriate traffic laws. Vehicles must yield the right of way appropriately at four-way stop signs, avoid various obstacles, and maintain a safe following distance.

The key difference between the Urban Challenge and the work in this thesis is that the Urban Challenge forces autonomous vehicles to work within current traffic laws and without any sort of explicit communication amongst vehicles or infrastructure. For the first few autonomous vehicles, this makes sense—most will have to deal with human drivers incapable of sending or receiving wireless transmissions. However, for a large population of autonomous vehicles, such etiquette and convention–based protocols waste a large portion of the vehicles' potential.

11.7 Safety Analysis

To the best of my knowledge, the failure mode analysis presented in this dissertation is the first study of the impact of this or any other such autonomous intersection protocol on driver safety. However, there is an enormous body of work regarding safety properties of traditional intersections. This includes the general—correlating traffic level and accident frequency [Sayed and Zein, 1999] and analyses of particular types of intersections [Bonneson and McCoy, 1993; Harwood *et al.*, 2003; Persaud *et al.*, 2001]—as well as plenty of more esoteric work, such as characterizing the role of Alzheimer's Disease in intersection collisions [Rizzo *et al.*, 2001]. However, because it concerns only human-operated vehicles, none of this work is particularly applicable to the setting with which this work is concerned.

In terms of managing and modelling the effects of incidents that have already happened, there is a large body of research. For instance, Boyles and Waller demonstrate that the severity of an incident can be measured in many ways, including the length of time to clear the incident, delays for other drivers, or even personal injury and property damage, and that accurate estimates of severity are crucial for maximizing the efficacy of limited resources for responding to the incident [Boyles and Waller, 2007]. Kamijo et al. have proposed an algorithm, called "spatio-temporal Markov random field" that tracks vehicles at intersections to determine when an

incident has occurred [Kamijo *et al.*, 2000]. Both of these, especially the latter, would be very relevant in a world of autonomous vehicles.

Chapter 12

Conclusion

This thesis presents a new mechanism for controlling automobile traffic at intersections. It also argues for a different approach to thinking about traffic problems—a multiagent approach in which vehicle is an independent and rational agent attempting to reach its destination as quickly as possible. While this approach may not be appropriate with traffic composed entirely of human-driven vehicles, for fully autonomous vehicles it offers many benefits over current methods. In the thesis, I have provided extensive empirical evidence to support this argument. This chapter first summarizes some of the more specific conclusions that can be drawn from this evidence. Second, this chapter briefly covers some of the limitations of the methods used to generate this evidence, namely experiments in simulation, as well as some suggestions of ways in which these limitations can be mitigated. Third, this chapter suggests some broader implications of this thesis's results in the realm of intelligent transportation systems. Last, I offer some promising future directions for this line of research.

12.1 Primary Conclusions

Intersections pose a critical challenge for autonomous vehicles, both in terms of safety and efficiency. Because vehicles at intersections are often traveling at such

high relative velocities (even on the freeway, most vehicles are not moving much with respect to one another), they are the most dangerous places for vehicles of any type, and the place where mistakes can have the gravest consequences. As such, I believe they are the best place to demonstrate the ways in which autonomous vehicles can make transportation both more efficient and safe.

This thesis has provided a protocol and some agent algorithms that demonstrate such a possibility in simulation. Most that have viewed videos of the `aim3` simulator have been amazed at what it accomplishes, while others have been more skeptical and demand to see real vehicles before they can accept the system. Nonetheless, this thesis takes a large, ambitious step toward such a system that can gain the acceptance of all but the most technophobic amongst us. Starting with the simplest of intersections and vehicles and scaling up to multiple complex intersections with a spectrum of vehicle types, this thesis demonstrates that autonomous vehicles have the capacity to greatly reduce the amount of time we waste in traffic, all the while making us more likely to arrive at our destinations in one piece.

12.2 Methodological Limitations

While this thesis addresses a wide gamut of issues, there are several limitations of the methodology used, mostly due to the fact that all the work was done in simulation. First, a real physical vehicle is much more complex than the model used in the simulator. It has three dimensions, the tires don't have perfect traction, and maximum acceleration is not constant over all velocities. While our simulated vehicles have the capability to experience sensor errors of many kinds, we have not yet explored the effects of those errors on safety or efficiency. These limitations are all valid concerns, but they are also an accepted part of a tradeoff we consciously made while pursuing this line of research. In exchange for losing some fidelity, we gain ease of implementation and agility of development. By not having to concern

ourselves with the exact acceleration profile of a real vehicle, we are free to try even more scenarios or agent algorithms. In terms of establishing a convincing case that multiagent technology has the capacity to improve automobile transportation and that it can be done safely, this thesis has accomplished its goals—the ability to adjust the safety buffers to account for deficiencies in the vehicle model allows me to make that claim.

There are several additional limitations that do not fall conveniently under the protection of our safety buffers, however. The first is that we have not extensively compared the performance of autonomous vehicles against realistic human drivers and control mechanisms. The system as designed is not exactly capable of such. A concern in its own right and a partial explanation for the previous concern, the simulator does not simulate all the complexities of real street layouts. Frequent driveways for businesses that cause vehicles to constantly enter and leave the roadway, dedicated turn lanes that appear only very near the intersection, and pedestrian traffic, are amongst the features of real-life driving that are not supported in our simulated environment. I must simply say that while these may have significant quantitative effects on the results, I don't believe they would alter the results' quality of supporting this mechanism's usefulness. It is left to future work to determine whether a more robust simulation environment or a test system with real vehicles is the best next step, although it will probably be some combination of the two.

One intersection configuration not discussed in this thesis is the roundabout. Popular in Europe, roundabouts do not require vehicles to stop, but rather to first merge onto the roundabout, and then exit onto an outbound lane. In some situations, roundabouts are highly efficient and very practical. However, in urban settings, roundabouts may not be an option, as they require a much larger area of land, which may not be available. Furthermore, vehicles must slow down for the roundabout, even in the total absence of other traffic, as merging onto the round-

about and turning around it cannot be done at as high speed as straight-line travel.

12.3 Future Directions

While this thesis comprises a lot of work on several important areas of autonomous intersection management, we have barely begun to understand the challenges associated with enormous populations of heterogeneous driverless vehicles. In this section, I briefly mention just a few of the possible next steps for this line of inquiry, including changing the nature of the agents, their algorithms, and even their setting.

12.3.1 Real Robots

The biggest step toward realizing autonomous intersection management will be getting it working on real vehicles. However, the cost of such an implementation is currently prohibitive for a small research group, especially considering the price of failures. As with similarly constrained technologies such as experimental aircraft, more simulation will certainly be prudent before attempting an all-physical implementation. But simulation and physical vehicles are not mutually exclusive! In a *mixed* simulation, real vehicles can be represented in the simulator via a *proxy vehicle*. That proxy vehicle can transmit simulated sensor readings (of other vehicles), while real actions taken by the physical vehicle are mapped back into the proxy vehicle (for instance, position or orientation updates). Most importantly, this works for even a single vehicle, in which case, we can be certain that no physical collisions will occur! Furthermore, a single-vehicle mixed simulation would still allow us to determine whether our algorithms for controlling the physical vehicle are correct and precise enough to move on to more physical vehicles. Such work has already begun, including a full-fledged mixed simulation with an autonomous vehicle that can traverse a real intersection while avoiding the other (simulated) vehicles [Nimmagadda, 2009]. Work using Denso Corporation's Wireless Safety Unit (WSU) [Denso

Corporation, 2006] has also started, in order to verify the feasibility of the protocol from Chapter 3 in a real-life situation [Beeson *et al.*, 2008].

12.3.2 Exploring Asynchronicity

The FCFS policy and its relatives presented in this thesis live up to the designation: "First Come, First Served." However, this property need not apply for every policy. Because the intersection control policy can process requests asynchronously, a policy could wait until a few requests have been received before making a decision as to which to accept and which to reject. This capability brings with it the question of how to choose which requests of a given set to approve and which to deny. Approaches from the simple—changing the order in which the requests are processed based on vehicle priority—to the complex—allowing vehicles to participate in a market- or auction-based system—represent a large unexplored space of possibilities.

12.3.3 Beyond Intersections

The view of autonomous traffic as a multiagent system is not new. In many other vehicle settings, such as freeway driving, research exists that tries to take advantage of autonomous vehicles. However, this thesis advocates eschewing modern traffic laws altogether in favor of protocols designed to maximize the utilization of autonomous vehicles' capabilities instead of merely adapting them to work with existing mechanisms. There are many situations in which artifacts of our human-centric systems might not be necessary or useful in systems comprising only autonomous vehicles.

For example, our modern roads are divided into lanes meant to accommodate all but the widest vehicles. Even when occupied by a very small vehicle, the entire lane is occupied. In a freeway situation, where the road may be five or six lanes wide, these extra allowances, aggregated over all the lanes, could be a source of waste. How often do six tractor-trailers pass one another simultaneously?

Parking lots are another such example. Parking lots are designed to store human-driven vehicles. As a result, every space must have direct access to an aisle, because the vehicles cannot move themselves out of the way when they are obstructing another vehicle's egress. This limitation is on top of the limitation that also dictates that all parking spaces must be of a minimum width (with the exception of the "compact" spaces that all-too-often contain a large sport-utility vehicle). It is not an infrequent occurrence for a motorist to begin pulling into a parking space only to discover that the entire space is taken up by a single motorcycle.

At a higher level, we may require fewer parking lots because autonomous vehicles make carpooling much easier. Taken to an extreme, a group of people could even share ownership of a vehicle, which could transport itself between uses by different people. There are many interesting questions regarding how to best share a fleet of vehicles amongst a population of drivers so as to minimize the travel time over all owners.

12.3.4 Beyond Automobiles

The algorithms and techniques in this thesis were directly inspired by the problem of controlling intersections of autonomous automobiles. In this problem, a scare resource (space-time in the intersection) must be allocated in such a way as to be useful to the consumers (vehicles). If a vehicle requires a certain region of space-time to cross the intersection, but can only reserve half of it, the utility of that space-time is not half of the total utility, but rather zero. The vehicle cannot cross. Furthermore, determining exactly what space-time is required by each vehicle may not be feasible without explicitly simulating the motion of the vehicle.

One problem that fits these characteristics is air traffic control. In air traffic control, the problem gets another dimension. Furthermore, an airplane cannot come to a stop to wait for airspace to free up. However, the limited resource (airspace-time) has the same nature, mostly due to the fact that one of its dimensions is time.

Airplanes move much more quickly than automobiles and can interact in much more complicated ways, such as causing localized disturbances in the surrounding air that can have drastic effects on other airplanes. However, given a large enough buffer around each airplane, the systems can be viewed as almost identical. Instead of intersection managers, airspace managers could be responsible for controlling access to various regions of airspace. Current air traffic control systems do not afford much, if any, autonomy to airplane pilots, but they are also not entirely under centralized control. Instead, each section of airspace is controlled by an individual human controller, who issues commands that pilots must then execute.

12.3.5 Policy Issues

In addition to the work that engineers, computer scientists, and mathematicians can do, many larger policy questions remain. Should such autonomous technology be mandated on new vehicles? Should human driving be allowed? How do we determine who is at fault when an incident, although unlikely, does occur? How does this technology affect automobile insurance premiums and policies? These questions must be left to those that can answer them: legislators and other policy makers, lawyers and courts, the insurance industry, and the automobile industry.

12.4 Broader Conclusions And Final Remarks

Autonomous vehicles are coming. Whether they are 10, 20, or 30 years away, the technology to create one exists and it is only a matter of time before the myriad benefits of computerized driver agents relegate the human-driven vehicle to novelty status. All other things being equal, the automation of the driving task will bring with it better fuel economy, more flexibility and freedom to passengers, and—most importantly—dramatically increased safety. But if this thesis argues anything, it is that all other things should not be left equal. Autonomous vehicles have capabilities that will be completely underutilized if they are forced to interact with one another

as current human drivers do. Mechanisms for controlling these vehicles must be rethought and reengineered with autonomous vehicles in mind.

This thesis presents such a rethinking of automobile traffic at intersections. In it, I have demonstrated that a multiagent mechanism for autonomous vehicle management can substantially decrease delays and accidents, thereby increasing the quality of life for travelers. One day soon, the simultaneously life-threatening and mundane task of driving automobiles will be in the capable hands of perpetually vigilant and disciplined software instead of absent-minded human drivers, who will in turn be free to talk on the phone, text message, or even sleep—without risk to themselves or others.

Appendix A

Glossary

This appendix contains a list of technical terms used throughout this thesis, along with definitions for each term.

active An active intersection manager is one that sends the EMERGENCY-STOP message when it detects a collision has occurred.

aim point The aim point is the point toward which the pilot turns the wheels to keep the vehicle in the current lane.

ACZ The ACZ, or Admission Control Zone, is an area beyond the intersection to which the intersection manager can control access.

capacity (ACZ) The capacity of an ACZ is the maximum total length of vehicles allowed in the ACZ at any time.

compatible Two trajectories through an intersection are compatible if they do not intersect.

conflict Two CLAIM messages are said to *conflict* if all of the following are true:

- The `intersection_id` fields of the two messages are identical.
- The paths determined by the `arrival_lane` and `departure_lane` fields are not *compatible* (compatible paths do not intersect).

- The time intervals are not disjoint.

coordinator The coordinator is the subagent of the driver agent which handles all of the vehicle's communication and interaction with other agents

crash log A crash log is a histogram of crashed vehicles. For a given amount of time after an incident, it contains the number of vehicles that have crashed since the time of the incident.

delay Delay is the amount of additional travel time incurred by a vehicle as the result of passing through the intersection. This may be directly due to the intersection or due to the indirect influence of the intersection through other vehicles

distance (ACZ) The ACZ distance is the length of the roadway leading away from the intersection over which the ACZ has control.

dominance Given two claims c_1 and c_2, we say that c_1 dominates c_2 if c_1 and c_2 conflict and c_1 has priority over c_2.

dominance graph The dominance graph G of a set of claims $C = \{c_1, c_2, \ldots, c_n\}$ is a digraph with vertices $V(G) = \{v_1, v_2, \ldots, v_n\}$, and directed edges $E(G) = \{(v_i, v_j) | c_i \text{ dominates } c_j\}$.

driver agent A driver agent is any agent that drives a vehicle. Usually, it means a computer program operating an autonomous vehicle.

edge tile An edge tile is a tile in an FCFS policy that resides in a part of the intersection where vehicles enter or leave.

following distance The following distance is the distance a vehicle keeps between itself and the vehicle in front of it.

frozen policy A frozen policy is a policy that is scheduled to be deactivated after a specified time. Reservations that would result in a vehicle being in the intersection after that time may not be accepted by a frozen policy.

granularity In an FCFS policy, the granularity is the length measurement of a side of a square reservation tile.

granularity ratio In an FCFS policy, the granularity ratio is defined to be $\frac{\sqrt{A}}{g}$, where A is the area of the intersection, and g is the granularity of the policy. For a square intersection broken into an $n \times n$ grid, the granularity ratio is n.

hybrid buffer A hybrid buffer is a buffer with both time and space components.

incident An incident is anything that prevents a vehicle from crossing an intersection according to its planned trajectory.

internal tile An internal tile is a reservation tile that is not an edge tile. Internal tiles are surrounded on all sides by reservation tiles.

intersection control policy An intersection control policy is a mechanism by which an intersection manager chooses to accept or reject reservation requests from approaching vehicles.

intersection manager The intersection manager is the agent controlling access to an intersection.

lead distance The lead distance is the distance from the projection of the vehicle's current position onto the vehicle's current lane to the vehicle's aim point.

lurk distance The distance from the intersection outside of which a coordinator will listen for other vehicles' V2V transmissions without sending anything.

lurking The behavior during which a vehicle listens for V2V transmissions without sending any of its own.

managed intersection A managed intersection is an intersection with an intersection manager—one in which an agent associated with that intersection governs access to the intersection.

mixed simulation A mixed simulation is a simulation in which certain proxy vehicles represent vehicles that are operating in the real world. Simulated sensor input from the simulator is provided to the real vehicle, and physical properties of the real vehicle are relayed back to the simulator to update the position of the proxy vehicle.

navigator The navigator is the component of the driver agent responsible for route planning and selection.

nonpermissible A nonpermissible CLAIM is one that is not guaranteed to be safe to follow in a V2V scenario.

oblivious An oblivious intersection manager is one that does not react when an incident has occurred.

off-limits tile An off-limits tile is a reservation tile that may not be reserved by autonomous vehicles.

omniscient An omniscient intersection manager is an intersection manager that knows at all times which intersection control policy best suits the needs of the current traffic.

optimistic An optimistic coordinator operates under the assumption that its vehicle will be able to accelerate without hindrance from other vehicles until it reaches the intersection.

passive A passive intersection manager is one that stops granting reservations after an incident has occurred, but does not send an EMERGENCY-STOP message to vehicles.

permissible A permissible CLAIM is one that is guaranteed to be safe to follow in a V2V scenario.

pessimistic A pessimistic coordinator operates under the assumption that it will not be able to accelerate beyond its current velocity before reaching the intersection.

pilot The pilot is the subagent of the driver agent responsible for physical manipulation of the vehicle.

priority Priority is a total, antisymmetric relation on CLAIM messages that allows driver agents to determine which vehicles may continue as planned and which should yield.

signal model A signal model is a predictive model of a real or simulated signal pattern for controlling vehicles at intersections.

traversal proposal A traversal proposal is a set of arrival and departure parameters that correspond to a proposed trajectory through an intersection.

reservation distance The reservation distance is a heuristic value designed to estimate the distance from the intersection of an approaching vehicle based on its reservation parameters.

reservation tile A reservation tile is a square corresponding to a physical square of intersection with an associated interval map from time intervals to vehicle identification numbers.

space buffer A space buffer is a buffer around a vehicle whose physical size is constant, always taking up the same amount of space.

time buffer A time buffer is a buffer around a vehicle that grows with the speed of the vehicle. It can be thought of as a buffer in the time dimension instead

of the space dimensions.

unmanaged intersection An unmanaged intersection is an intersection without an intersection manager.

V2I A vehicle-to-infrastructure (or -intersection) scenario is one in which vehicles communicate with an intersection manager stationed at the intersection.

V2V A vehicle-to-vehicle or V2V scenario is one in which vehicles communicate solely with other vehicles.

Bibliography

[Abdulhai *et al.*, 2003] Baher Abdulhai, Rob Pringle, and Grigoris J. Karakoulas. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285, May/June 2003.

[Alvarez and Horowitz, 1997] Luis Alvarez and Roberto Horowitz. Traffic flow control in automated highway systems. Technical Report UCB-ITS-PRR-97-47, University of California, Berkeley, Berkeley, California, USA, November 1997.

[Balan and Luke, 2006] Gabriel Balan and Sean Luke. History-based traffic control. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 616–621, Hakodate, Japan, May 2006.

[Bazzan, 2005] Ana L. C. Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10(2):131–164, March 2005.

[Beeson *et al.*, 2008] Patrick Beeson, Jack O'Quin, Bartley Gillan, Tarun Nimmagadda, Mickey Ristroph, David Li, and Peter Stone. Multiagent interactions in urban driving. *Journal of Physical Agents*, 2(1):15–30, March 2008. Special issue on Multi-Robot Systems.

[Bishop, 2005] Richard Bishop. *Intelligent Vehicle Technology and Trends*. Artech House, 2005.

[Bonneson and McCoy, 1993] J. A. Bonneson and P. T. McCoy. Estimation of safety at two-way stop–controlled intersections on rural highways. *Transportation Research Record*, 1401:83–89, 1993.

[Boyles and Waller, 2007] Stephen Boyles and S. Travis Waller. A stochastic delay prediction model for real-time incident management. 77:18–24, April 2007.

[Bull *et al.*, 2004] L. Bull, J. Sha'Aban, A. Tomlinson, J. D. Addison, and B. G. Heydecker. Towards distributed adaptive control for road traffic junction signals using learning classifier systems. In L. Bull, editor, *Applications of Learning Classifier Systems*, pages 276–299. Springer, 2004.

[Caliper Corporation, 2009] Caliper Corporation. Transmodeler, 2009. `http://www.caliper.com/transmodeler/Simulation.htm`.

[Clement, 2002] Stuart Clement. The SPA model with smooth acceleration. In *24th Conference of Australian Institutes of Transport Research (CAITR-2002)*, Sydney, Australia, December 2002.

[DARPA, 2007a] DARPA. The DARPA urban challenge, 2007. `http://www.darpa.mil/grandchallenge`.

[DARPA, 2007b] DARPA. Route network definition file (rndf) and mission data file (mdf) formats, March 2007. Accessed online at `http://www.darpa.mil/GRANDCHALLENGE/docs/RNDF_MDF_Formats_031407.pdf`.

[Denso Corporation, 2006] Denso Corporation. Wireless safety unit (WSU). `http://www.denso.co.jp/en/events/globalmotorshows/download/itswc06/pdf/%wirelesssafetyunit.pdf`, 2006. Accessed 5/26/2009.

[Drabkin *et al.*, 2007] Vadim Drabkin, Roy Friedman, Gabriel Kliot, and Marc Segal. Rapid: Reliable probabilistic dissemination in wireless ad-hoc networks. In

The 26th IEEE International Symposium on Reliable Distributed Systems, Beijing, China, October 2007.

[Dresner and Stone, 2004] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 530–537, New York, NY, USA, July 2004.

[Dresner and Stone, 2005] Kurt Dresner and Peter Stone. Multiagent traffic management: An improved intersection control mechanism. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477, Utrecht, The Netherlands, July 2005.

[Dresner and Stone, 2006] Kurt Dresner and Peter Stone. Multiagent traffic management: Opportunities for multiagent learning. In K. Tuyls et al., editor, *LAMAS 2005*, volume 3898 of *Lecture Notes In Artificial Intelligence*, pages 129–138. Springer Verlag, Berlin, 2006.

[Fischer *et al.*, 1985] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, January 1979.

[Gavrila *et al.*, 2004] D. M. Gavrila, J. Giebel, and S. Munder. Vision-based pedestrian detection: The PROTECTOR+ system. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2004)*, Parma, Italy, June 2004.

[Gepperth *et al.*, 2005] Alexander Gepperth, Johann Edelbrunner, and Thomas Bücher. Real-time detection and classification of cars in video sequences. In

Proceedings of the IEEE Intelligent Vehicle Symposium (IV2005), pages 625–631, Las Vegas, NV, USA, June 2005.

[Hallé and Chaib-draa, 2005] Simon Hallé and Brahim Chaib-draa. A collaborative driving system based on multiagent modelling and simulations. *Journal of Transportation Research Part C (TRC-C): Emergent Technologies*, 13:320–345, 2005.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. 4:100–107, July 1968.

[Harwood *et al.*, 2003] Douglas W. Harwood, Karin M. Bauer, Ingrid B. Potts, Darren J. Torbic, Karen R. Richard, Emilia R. Kohlmann Rabbani, Ezra Hauer, Lily Elefteriadou, and Michael S. Griffith. Safety effectiveness of intersection left- and right-turn lanes. *Transportation Research Record*, 1840:131–139, 2003.

[Hatipo *et al.*, 1997] Cem Hatipo, Keith Redmill, and Umit Ozguner. Steering and lane change: A working system. In *IEEE Conference on Intelligent Transportation Systems*, pages 272–277, November 1997.

[Helbing *et al.*, 2001] Dirk Helbing, Ansgar Hennecke, Vladimir Shvetsov, and Martin Treiber. MASTER: Macroscopic traffic simulation based on a gas-kinetic, non-local traffic model. *Transportation Research Part B: Methodological*, 35(2):183–211, February 2001.

[Hunt *et al.*, 1981] P B Hunt, D I Robertson, R D Bretherton, and R I Winton. SCOOT - a traffic responsive method of co-ordinating signals. Technical Report TRRL-LR-1014, Transport and Road Research Laboratory, 1981.

[Johnson, 2005] R. Colin Johnson. Steady pace takes DARPA race. EE Times, October 2005. Accessed at `http://www.eetimes.com`.

[Kamijo *et al.*, 2000] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ijeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):108–118, June 2000.

[Kohl *et al.*, 2006] Nate Kohl, Kenneth Stanley, Risto Miikkulainen, Michael Samples, and Rini Sherony. Evolving a real-world vehicle warning system. In *Proceedings of the Genetic and Evolutionary Computation Conference 2006*, Seattle, WA, USA, July 2006.

[Kolodko and Vlacic, 2003] Julian Kolodko and Ljubo Vlacic. Cooperative autonomous driving at the intelligent control systems laboratory. *IEEE Intelligent Systems*, 18(4):8–11, July/August 2003.

[Laumônier *et al.*, 2006] Julien Laumônier, Charles Desjardins, and Brahim Chaibdraa. Cooperative adaptive cruise control: a reinforcement learning approach. In *The Fourth Workshop on Agents in Traffic and Transportation*, Hakodate, Hokkaido, Japan, May 2006.

[Lindner *et al.*, 2004] F. Lindner, U. Kressel, and S. Kaelberer. Robust recognition of traffic signals. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2004)*, Parma, Italy, June 2004.

[Liu and Fujimura, 2003] Xia Liu and Kikuo Fujimura. Pedestrian detection using stereo night vision. In *IEEE International Conference on Intelligent Transportation Systems*, Shanghai, China, October 2003.

[Mählisch *et al.*, 2005] Mirko Mählisch, Matthias Oberländer, Otto Löhlein, Dariu Gavrila, and Werner Ritter. A multiple detector approach to low-resolution FIR pedestrian recognition. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2005)*, Las Vegas, NV, USA, June 2005.

[Mirchandani and Wang, 2005] Pitu Mirchandani and Fei-Yue Wang. Rhodes to intelligent transportation systems. *IEEE Intelligent Systems*, 20(1):10–15, 2005.

[Moriarty and Langley, 1998] David Moriarty and Pat Langley. Learning cooperative lane selection strategies for highways. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 684–691, Madison, WI, 1998. AAAI Press.

[National Highway Traffic Safety Administration, 2002] National Highway Traffic Safety Administration. Economic impact of U.S. motor vehicle crashes reaches $230.6 billion, new NHTSA study shows. NHTSA Press Release 38-02, May 2002. `http://www.nhtsa.dot.gov`.

[Naumann and Rasche, 1997] Rolf Naumann and Rainer Rasche. Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles. In *Proceedings of the 30th ISATA - ATT/IST Conference*, 1997.

[Naumann *et al.*, 1998] Rolf Naumann, Rainer Rasche, and Jürgen Tacken. Managing autonomous vehicles at intersections. *IEEE Intelligent Systems*, 13(3):82–86, May 1998.

[Nimmagadda, 2009] Tarun Nimmagadda. Building an autonomous ground traffic system. Technical Report HR-09-09, The University of Texas at Austin, August 2009.

[Noda *et al.*, 2006] Itsuki Noda, Adam Jacoff, Ansgar Bredenfeld, and Yasutake Takahashi, editors. *RoboCup-2005: Robot Soccer World Cup IX*. Springer Verlag, Berlin, 2006.

[NOVA, 2006] NOVA. The great robot race, 2006. Originally aired 28 March 2006 on PBS, available online at `http://www.pbs.org/wgbh/nova/darpa`.

[Pease *et al.*, 1980] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[Persaud *et al.*, 2001] Bhagwant N. Persaud, Richard A. Retting, Per E. Gardner, and Dominique Lord. Safety effect of roundabout conversions in the united states: Empirical bayes observational before-after study. *Transportation Research Record*, 1751:1–8, 2001.

[Pomerleau, 1993] Dean A. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, 1993.

[Ramström and Christensen, 2005] Ola Ramström and Henrik Christensen. A method for following umarked roads. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV2005)*, pages 650–655, Las Vegas, NV, USA, June 2005.

[Rasche *et al.*, 1997] R. Rasche, R. Naumann, J. Tacken, and C. Tahedl. Validation and simulation of decentralized intersection collision avoidance algorithm. In *Proceedings of IEEE Conference on Intelligent Transportation Systems (ITSC 97)*, 1997.

[Reece and Shafer, 1991] Douglas A. Reece and Steven Shafer. A computational model of driving for autonomous vehicles. Technical Report CMU-CS-91-122, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, April 1991.

[Reynolds, 1999] Craig W. Reynolds. Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference*, pages 763–782, 1999.

[Rizzo *et al.*, 2001] Matthew Rizzo, Daniel V. McGehee, Jeffrey D. Dawson, and Steven N. Anderson. Simulated car crashes at intersections in drivers with Alzheimer disease. *Alzheimer Disease and Associated Disorders*, 15(1):10–20, 2001.

[Robertson, 1969] D I Robertson. TRANSYT — a traffic network study tool. Technical Report TRRL-LR-253, Transport and Road Research Laboratory, 1969.

[Rogers *et al.*, 1999] Seth Rogers, Claude-Nicolas Flechter, and Pat Langley. An adaptive interactive agent for route advice. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 198–205, Seattle, WA, USA, 1999. ACM Press.

[Roozemond, 1999] Danko A. Roozemond. Using intelligent agents for urban traffic control systems. In *Proceedings of the International Conference on Artificial Intelligence in Transportation Systems and Science*, pages 69–79, 1999.

[Sayed and Zein, 1999] Tarek Sayed and Sany Zein. Traffic conflict standards for intersections. *Transportation Planning and Technology*, 22(4):309–323, August 1999.

[Schonberg *et al.*, 1995] T. Schonberg, M. Ojala, J. Suomela, A. Torpo, and A. Halme. Positioning an autonomous off-road vehicle by using fused DGPS and inertial navigation. In *2nd IFAC Conference on Intelligent Autonomous Vehicles*, pages 226–231, 1995.

[She *et al.*, 2004] Kai She, George Bebis, Haisong Gu, and Ronald Miller. Vehicle tracking using on-line fusion of color and shape features. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, Washington, DC, USA, October 2004.

[Sukthankar *et al.*, 1995] Rahul Sukthankar, Dean Pomerleau, and Chuck Thorpe. SHIVA: Simulated highways for intelligent vehicle algorithms. In *Proceedings of the 1995 IEEE Intelligent Vehicles Symposium*, 1995.

[Svenson, 1981] Ola Svenson. Are we all less risky and more skillful than our fellow drivers? *Acta Psychologica*, 47(2):143–148, February 1981.

[Texas Transportation Institute, 2004] Texas Transportation Institute. 2004 urban mobility report, September 2004. Accessed at `http://mobility.tamu.edu/ums` in December 2004.

[Turner, 1986] J.S. Turner. New directions in communications (or which way to the information age?). *Communications Magazine, IEEE*, 24(10):8–15, October 1986.

[USDOT, 2003] USDOT. Inside the USDOT's 'intelligent intersection' test facility. Newsletter of the ITS Cooperative Deployment Network, July 15, 2003. Accessed online 17 May 2006 at `http://www.ntoctalks.com/icdn/intell_intersection.php`.

[VanMiddlesworth *et al.*, 2008] Mark VanMiddlesworth, Kurt Dresner, and Peter Stone. Replacing the stop sign: Unmanaged intersection control for autonomous vehicles. In *The Fifth Workshop on Agents in Traffic and Transportation*, pages 94–101, Estoril, Portugal, May 2008.

[Watanabe and Nishida, 2005] A. Watanabe and M. Nishida. Lane detection for a steering assistance system. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV2005)*, pages 159–164, Las Vegas, NV, USA, June 2005.

[Wiering, 2000] M. A. Wiering. Multi-agent reinforcement learning for traffic light control. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'2000)*, pages 1151–1158, 2000.

[Wierwille *et al.*, 2002] W. W. Wierwille, R. J. Hanowski, J. M. Hankey, C. A. Kieliszewski, S. E. Lee, A. Medina, A. S. Keisler, and T. A. Dingus. Identification and evaluation of driver errors: Overview and recommendations. Technical

Report FHWA-RD-02-003, Virginia Tech Transportation Institute, Blacksburg, Virginia, USA, August 2002. Sponsored by the Federal Highway Administration.

[Witten and Frank, 2005] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[Wooldridge, 2002] Michael Wooldridge. *Introduction to MultiAgent Systems*. John Wiley & Sons, February 2002.

[Wu *et al.*, 2005] Shinq-Jen Wu, Hsin-Han Chiang, Jau-Woei Perng, Tsu-Tian Lee, and Chao-Jung Chen. The automated lane-keeping design for an intelligent vehicle. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV2005)*, pages 508–513, Las Vegas, NV, USA, June 2005.

Vita

Kurt Dresner was born at Fort Lewis, Washington, as the son of two career Army officers, and grew up living in Hawaii with his father. He graduated from University High School in Tucson, Arizona in 1998. He attended Harvey Mudd College in Claremont, California, where he graduated in 2002 *with distinction*, earning a Bachelor of Science degree in computer science and mathematics, earning honors from both departments. That fall, he enrolled in the PhD program in the Department of Computer Sciences at the University of Texas at Austin, where he was a teaching assistant, graduate research assistant, and an assistant instructor. He founded the Autonomous Intersection Management project with his advisor, Peter Stone, which received grant support from General Motors and the Federal Highway Administration. Kurt is currently employed as a software engineer by Google, Inc. at the Kirkland, Washington campus.

Permanent Address: 11224 NE 61st Place, Kirkland, WA 98033 USA

kdresner@cs.utexas.edu

This dissertation was typeset with LaTeX 2$_\varepsilon$[1] by the author.

[1] LaTeX 2$_\varepsilon$ is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended

by Bert Kay, James A. Bednar, and Ayman El-Khashab.